Chapter 1

인스턴스 : 지금까지 배웠던 변수를 객체라는 다른 형식으로 다루는 것

auto : 초깃값의 자료형에 맞춰 선언하는 인스턴스의 자료형을 '자동으로' 결정합니다.

참조형 변수 : 지긋지긋했던 포인터의 오류를 줄여주는 자료 다루기 방법

범위 기반 for문 : 특정 조건 없이도 반복문을 사용하도록 도와줍니다.

입 · 출력 예제

```
#include "pch.h"        // 미리 컴파일된 헤더
#include <iostream>     // == <stdio.h>

int main()
{
    std::cout << "Hello World!" << std::endl;    // std-소속,{ ::-범위 지정 연산자, cout-콘솔 출력 담당 객체, <<-연산자 함수
    return 0;
}
```

```
/* std::cout 사용법 */
#include "pch.h"
#include <iostream>

int main()
{
    std::cout << 10 << std::endl;
    std::cout << 10U << std::endl;
    std::cout << 10.5F << std::endl;
    std::cout << 10.5 << std::endl;
    std::cout << 3 + 4 << std::endl;

    return 0;
}
```

```
/* 문자열 조합 출력 */
#include "pch.h"
#include <string>
#include <iostream>

int main()
{
    std::string strData = "Test string";
    std::cout << "Sample string" << std::endl;
    std::cout << strData << std::endl;

    strData = "New string";
    std::cout << strData << std::endl;

    std::cout << "저는 " << 20 << "살" << "입니다." << std::endl;

    return 0;
}
```

```
/* std::cin 사용법 */
#include "pch.h"
#include <string>
#include <cstdio>
#include <iostream>

int main()
{
    int nAge;
    std::cout << "나이를 입력하세요." << std::endl;
    std::cin >> nAge;

    char szJob[32];
    std::cout << "직업을 입력하세요." << std::endl;
    std::cin >> szJob;

    std::string strName;
    std::cout << "이름을 입력하세요." << std::endl;
    std::cin >> strName;

    std::cout << "당신의 이름은 " << strName << "이고, "
        << "나이는 " << nAge << "살이며, "
        << "직업은 " << szJob << "입니다." << std::endl;

    return 0;
}
```

자료형

자료형	설명
long long	64비트 정수(컴파일러에 따라 약간 다를 수 있음)
char16_t	16비트 문자(ex. char16_t a = u’A’;)
char32_t	32비트 문자(ex. char32_t a = u’A’;)
auto	컴파일러가 자동으로 형식을 규정하는 자료형(ex. auto a =10;)
decltype(expr)	expr과 동일한 자료형(ex. int x = 10;decltype(x) y = 20;)

```
/* auto 예약어 사용 */
#include "pch.h"
#include <iostream>

int main(void)
{
    int a = 10;
    int b(a);
    auto c(b);    // 초깃값의 형식에 맞춰 선언하는 인스턴스의 형식이 '자동'으로 결정

    std::cout << a + b + c << std::endl;

    return 0;
}
```

메모리 동적 할당

new : 객체를 동적 할당하는 ‘연산자’ (malloc ( )과 다른 점 : 메모리의 크기를 정하지 않는다.

new 연산자 : 객체의 생성자 호출 , delete 연산자 : 객체의 소멸자 호출)

delete : 객체를 동적 해제하는 ‘연산자’

```
/* new 연산자 사용 */
#include "pch.h"
#include <iostream>

int main()
{
    // 인스턴스만 동적으로 생산하는 경우
    int* pData = new int;

    // 초깃값을 기술하는 경우
    int* pNewData = new int(10);

    *pData = 5;
    std::cout << *pData << std::endl;
    std::cout << *pNewData << std::endl;

    delete pData;
    delete pNewData;
}
```

```
/* 배열 형태의 객체 생성 */
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    // 객체를 배열 형태로 동적 생산한다.
    int* arr = new int[5];
    for (int i = 0; i < 5; ++i)
        arr[i] = (i + 1) * 10;

    for (int i = 0; i < 5; ++i)
        cout << arr[i] << endl;

    // 배열 형태로 생성한 대상은 반드시 배열 형태를 통해 삭제한다
    delete[] arr;

    return 0;
}
```

참조자

“상수에는 참조자를 선언할 수 없다”

가능

불가능

int &rData = a;

int *pData = &3;

int &rData = 3;

int &rData;

```
/* 참조형 변수 사용 */
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    int nData = 10;

    // nData 변수에 대한 참조자 선언
    int& ref = nData;

    // 참조자의 값을 변경하면 원본도 변경된다!
    ref = 20;
    cout << nData << endl;

    // 포인터를 쓰는 것과 비슷하다.
    int* pnData = &nData;
    *pnData = 30;
    cout << nData << endl;

    return 0;
}
```

“덩치 큰 자료는 값이 아니라 ‘주소’를 전달하는 것이 효율적이다!”

```
/* 참조에 의한 호출 */
#include "pch.h"
#include <iostream>
using namespace std;

// 매개변수가 int에 대한 참조 형식이다.
void TestFunc(int& rParam)
{
    // 피호출자 함수에서 원본의 값을 변경했다.
    rParam = 100;
}

int main()
{
    int nData = 0;

    // 참조에 의한 인수 전달이다.
    TestFunc(nData);
    cout << nData << endl;

    return 0;
}
```

“호출자 코드만 봐서는 참조 형식인지 모른다”

```
/* 참조 전달 */
#include "pch.h"
#include <iostream>
using namespace std;

// 참조 전달이므로 호출자 변수의 값을 변경할 수 있다.
void Swap(int& a, int& b)
{
```

```
    int nTmp = a;
    a = b;
    b = nTmp;
}

int main()
{
    int x = 10, y = 20;

    // 참조 전달이며 두 변수의 값이 교환된다.
    Swap(x, y);

    // 교환된 결과를 출력한다.
    cout << "x: " << x << endl;
    cout << "y: " << y << endl;

    return 0;
}
```

포인터 대신 참조!!

```
/* r-value 참조 */
#include "pch.h"
#include <iostream>
using namespace std;

int TestFunc(int nParam)
{
    int nResult = nParam * 2;

    return nResult;
}

int main()
{
    int nInput = 0;
    cout << "Input number: ";
    cin >> nInput;

    // 산술 연산으로 만들어진 임시 객체에 대한 r-value 참조
    int&& rdata = nInput + 5;
    cout << rdata << endl;

    // 함수 반환으로 만들어진 임시 객체에 대한 r-value 참조
    int&& result = TestFunc(10);

    // 값을 변경할 수 있다.
    result += 10;
    cout << result << endl;

    return 0;
}
```

‘임시 결과는 이어지는 연산에 활용된 직후 소멸하는 r-value’

범위 기간 for문

```
/* 범위 기반 for문 */
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    int aList[5] = { 10,20,30,40,50 };
```

```
    // 전형적인 C 스타일 반복문
    for (int i = 0; i < 5; ++i)
        cout << aList[i] << ' ';

    cout << endl;

    // 범위 기반 C++11 스타일 반복문
    // 각 요소의 값을 n에 복사한다.
    for (auto n : aList)
        cout << n << ' ';

    cout << endl;

    // n은 각 요소에 대한 참조다.
    for (auto& n : aList)
        cout << n << ' ';

    cout << endl;

    return 0;
}
```

for(auto &n : aList)처럼 참조자를 요소 형식으로 선언

문제

이름과 나이 입력받기

```
/* 문제 1 */
#include "pch.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int age;
    string name;
    cout << "이름 : " << endl;
    cin >> name;
    std::cout << "나이 : " << endl;
    std::cin >> age;
    cout << "나의 이름은 " << name << "이고, 나이는 " << age << "살입니다." << endl;
    return 0;
}
```

auto의 의미와 예

자료형을 자동으로 맞춰줌

```
/* 문제2 */
#include "pch.h"
#include <iostream>
using namespace std;

int main() {
    auto a(10);
    cout << a << endl;
    return 0;
}
```

new 연산자 동적 할당하고 해제

```
/* 문제3 */
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    char* arr = new char[12];
    delete[] arr;
    return 0;
}
```

Swap

```
/* 문제4 */
#include "pch.h"
#include <iostream>
using namespace std;

void Swap(int& a, int& b)
{
    int tmp = a;
    a = b;
    b = tmp;
}

int main()
{
    int x = 10, y = 50;
    Swap(x, y);
    cout << "x: " << x << ' ' << "y: " << y << endl;
```

```
    return 0;
}
```

상수형 참조와 기존 참조 형식의 다른 점

상수에는 참조를 못함

int 배열을 오름차순 후 출력

```
/* 문제6 */
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    int aList[5] = { 30,20,50,10,40 };
    int tmp;

    for (int j = 0; j < 5; j++) {
        int min = 100;
        int num = 0;
        for (int i = j; i < 5; i++) {
            if (min > aList[i]) {
                min = aList[i];
                num = i;
            }
        }
        tmp = aList[num];
        aList[num] = aList[j];
        aList[j] = tmp;
    }
    cout << "int aList[5] = { ";
    for (auto n : aList)
        cout << n << ", ";
    cout << "};";
    return 0;
}
```