TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

2018-06-07 (69 회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경
ucong@naver.com

C++ C 언어 사용 + 개발속도 내고 싶을때 사용한다.

ex. 게임회사는 대부분 C++ 쓴다, 영상처리, 임베디드

cf. 커널은 C 나 어셈을 사용한다

cf. C, 파이썬, 자바 합친 언어가 go

확장자: .cpp

컴파일: g++ -g(디버깅옵션) [파일명.cpp] -o [실행파일명]

1. hello world 출력

```
#include <iostream> // stdio.h 와 동일
using std::cout;
using std::endl;

int main(void)
{
        cout << "Hello World!!" << endl; // cout: print // endl: 개행

        cout << "Hello" << " World!!" << endl;
        cout << 1 << 'a' << " String"  << endl;

        return 0;
}
```

2. 입력받은 값으로 연산하기

```
#include <iostream>

using std::cout;

using std::endl;

using std::cin; /*cin: canf , read 0 번*/


int main(void){
        int val1, val2;

        cout << "1 번째 정수 입력: ";

        cin >> val1;

        cout << "2 번째 정수 입력: ";

        cin >> val2;

        int result = val1 + val2;

        cout << "덧셈결과: " << result << endl;

        return 0;

}
```

3. 함수를 이용한 연산

```
#include <iostream>

using std::cout;

using std::endl;


int function(void){ return 10; }

int function(int a, int b){ return a+b;}


int main(void){
        int result;

        cout << function() << endl;

        cout << function(7,77) << endl;

        return 0;

}
```

4. function 안에 비어있는 데 자동으로 채워진다

```
#include <iostream>

using std::cout;

using std::endl;

int function(int a = 0){ return a + 1; }

int main(void)

{
        int result;

        cout << function() << endl;

        cout << function(7) << endl;

        return 0;

}
```

5. Inline 이랑 #define 이랑 같은 코드

```
#include <iostream>

using std::cout;

using std::endl;

// #define SQUARE() ((x)*(x)) 아래와 같은 코드

inline int SQUARE(int x) { return x*x; }

int main (void){
        int result;

        cout << SQUARE(5) << endl;

        return 0;

}
```

기계어 분석하면 함수 호출시 스택프레임 생성하고 해제하는데

인라인 붙여보면 그 과정(함수 호출시 스택호출을 안함)이 없어짐

cf. 함수 크기가 크면 인라인이 의미가 없어짐

인라인 함수: C/C++ 언어에서 사용할 수 있는 기능으로, 함수 호출 시 별도로 분리된 위치의 레이블로 점프하여 실행되는 일반 함수와는 달리 호출 부분을 함수 전체 코드로 치환하여 컴파일한다.

사용방법은 무척 심플한데, inline 을 함수 선언시 붙여주면 된다.

https://namu.wiki/w/%EC%9D%B8%EB%9D%BC%EC%9D%B8%20%ED%95%A8%EC%88%98

6. 네임 스페이스를 구분해 놓으면 이름이 같아도 구동이 된다

```
#include <iostream>

using std::cout;

using std::endl;

namespace A{
        void test(void){ cout << "A 에서 정의한 함수" << endl; }
}

namespace B{
        void test(void){ cout << "B 에서 정의한 함수" << endl; }
```

```
}


int main(void){
        A::test();
        B::test();
        return 0;
}
```

이름공간 또는 **네임스페이스**(Namespace)는 개체를 구분할 수 있는 범위를 나타내는 말로 일반적으로 하나의 이름 공간에서는 하나의 이름이 단 하나의 개체만을 가리키게 된다.

컴퓨터 프로그래밍 언어인 C 에서는 전역 지역 공간과 지역 이름 공간라는 이름 공간에 대한 개념이 있는데, 각각의 이름 공간에서는 같은 변수나 함수 이름을 사용할 수 없지만, 영역이 다르면 변수나 함수명이 같을 수도 있다. C++와 Java 프로그래밍 언어에서는 이름 공간을 명시적으로 지정하여 사용할 수 있다.

https://ko.wikipedia.org/wiki/%EC%9D%B4%EB%A6%84%EA%B3%B5%EA%B0%84

7. ref : 변수의 별명을 만든다 (포인터와 유사하지만 포인터와 다르게 NULL 값으로 만들수는 없다)

```
#include <iostream>

using namespace std;

int main (void){
        int val = 10;
        int &ref = val;

        val++;
        cout << "ref:" << ref << endl;
        cout << "val:" << val << endl;
        ref++;
        cout << "ref:" << ref << endl;
        cout << "val:" << val << endl;
        return 0;
}
```

http://simplesolace.tistory.com/438

reference 는 변수의 별명이다.

선언방법: int &[ref: 별명을 적어준다] = val;

여기서 val 의 주소와 ref 의 주소가 같다

레퍼런스를 가지고 하는 연산은 레퍼런스가 참조하는 변수의 이름을 가지고 하는 연산과 같은 효과를 지닌다.

포인터랑 비슷하지만, 포인터처럼 널값을 셋팅할 수는 없다

C 언어에는 Call by value 뿐

C++ 에서 Call by reference 가 가능하다. (다음 예제 참고)

*. 래퍼런스는 객체에 대한 다른 이름이다.

http://egloos.zum.com/dstein/v/2949479

8. call by reference 예제 swap()

```
#include <iostream>

using namespace std;

void swap(int& a, int& b){
        int temp = a;
        a = b; b = temp;
}

int main(void){
        int val1 = 10;
        int val2 = 20;
        cout << "val1:" << val1 << ' ';
        cout << "val2:" << val2 << endl;
        swap(val1, val2);
        cout << "val1:" << val1 << ' ';
        cout << "val2:" << val2 << endl;
        return 0;
}
```

Assembly Analysis 를 보니 포인터랑 똑같다.

(gdb a.out → b main → r → n → disas)

```
Dump of assembler code for function swap(int&, int&):
   0x0000000000400976 <+0>:     push   %rbp
   0x0000000000400977 <+1>:     mov    %rsp,%rbp
   0x000000000040097a <+4>:     mov    %rdi,-0x18(%rbp)
   0x000000000040097e <+8>:     mov    %rsi,-0x20(%rbp)
=> 0x0000000000400982 <+12>:    mov    -0x18(%rbp),%rax
   0x0000000000400986 <+16>:    mov    (%rax),%eax
   0x0000000000400988 <+18>:    mov    %eax,-0x4(%rbp)
   0x000000000040098b <+21>:    mov    -0x20(%rbp),%rax
   0x000000000040098f <+25>:    mov    (%rax),%edx
   0x0000000000400991 <+27>:    mov    -0x18(%rbp),%rax
   0x0000000000400995 <+31>:    mov    %edx,(%rax)
   0x0000000000400997 <+33>:    mov    -0x20(%rbp),%rax
   0x000000000040099b <+37>:    mov    -0x4(%rbp),%edx
   0x000000000040099e <+40>:    mov    %edx,(%rax)
   0x00000000004009a0 <+42>:    nop
   0x00000000004009a1 <+43>:    pop    %rbp
   0x00000000004009a2 <+44>:    retq
```

9. 배열 동적할당

(malloc 에 해당하는 녀석은 new / free 에 해당하는 녀석은 delete)

```
#include <iostream>

using namespace std;

int main(void) {
        int size;
        cout << "할당하고자 하는 배열의 크기:";
        cin << size;
        int *arr = new int[size];
        for(int i = 0; i <size; i++)
        arr[i] = i+1;
        for(int j = 0; j <size; j++)
                cout << "arr[" << j << "]=" << arr[j] <<endl;
        delete []arr;
        return 0;
}
```

https://blog.naver.com/cestlavie_01/221144770617

cf. c 에서 할때는

```
stack *tmp;
tmp = (stack*) malloc(sizeof(stack));
```

10. Abstraction(추상화) 이란? C++면접 문제

C 언어에서 구조체 안에 해당 구조체와 관련된 함수포인터를 같이 넣으면 객체 관리하기가 매우 쉽다.

구조체에서 조금 더 발전된 개념이 C++에서의 Class 이다.

사용할 Data(변수), Method(함수, 기능)를 생각하고 이를 Class 화 하는 것을 추상화라고 한다.

1) 코드가 매우 길다. 만줄 이상일때 구조체 안에 변수와 함수들 넣어두면

구조체만 보아도 이 변수들이 함수들을 제어한다는 것을 알 수 있다.

이게 바로 추상화이다. 사용하는 변수들과 그 변수들을 사용하는 함수가 같은 구조체 안에 묶여 있어야 한다.

2) 클래스에 접근 못하게 하기 위해 Public, Private 두가지가 있다.

은행 고객 계좌 관리하는 변수를 실수로 건드리는 일이 없다.

위의 두가지를 적절히 섞어서 추상화를 구현한다.

https://kldp.org/node/64458

클래스 내에서 멤버의 접근 허용범위를 명시하는 Public(어디에서든 접근 허), Protected, Private(클래스 내부에서만 접근 허용) 가 있다.

함수는 public

변수는 private 이라고 쓰는게 좋은 추상화

→ 신입 개발자가 변수명에 직접 접근해서 계좌를 0 으로 만들어버리는 일이 없도록

반드시 함수를 사용해서 접근하도록 한다 (변수명에 직접 지정하면 오류발생)

11. 추상화가 잘된 코드 (메서드를 거쳐야 private 변수에 접근이 가능하다)

```
#include <iostream>

using namespace std;

const int OPEN = 1;

const int CLOSE = 2;

class Door{

private:

        int state;

public:

        void Open();

        void Close();

        void ShowState();

};

void Open(void){ state = OPEN; }

void Close(void){ state = CLOSE; }

void ShowState(void){

        cout << "현재 문의 상태:";

        cout << ((state == OPEN)? "OPEN" : "CLOSE") << endl;

}


int main(void){

        Door door;

        door.Open();

        door.ShowState();

        door.Close();

        door.ShowState();

        return 0;

}
```

12. DSP 사용시 코드가 길어서 파일분할을 자주 하게된다.

위의 11 번 코드에서 헤더파일을 따로 작성하는 연습을 해보자

세개의 코드를 작성한다.

Door.h

```
#include <iostream>
using namespace std;
const int OPEN =1;
const int CLOSE =2;
class Door{
private:
        int state;
public:
        void Open();

        void Close();

        void ShowState();

};
```

main.cpp

```
#include “Door.h”

int main(void){

        Door door;

        door.Open();

        door.ShowState();

        door.Close();

        door.ShowState();

        return 0;

}
```

Door.cpp

```
#include “Door.h”

void Open(void){ state = OPEN; }

void Close(void){ state = CLOSE; }

void ShowState(void){

        cout << "현재 문의 상태:";

        cout << ((state == OPEN)? "OPEN" : "CLOSE") << endl;

}
```

다음과 같이 컴파일 한다.

g++ Door.cpp Main.cpp -o [실행파일명] 컴파일하고 ./[실행파일명]

13. [Quiz 4] 성적관리 Class 를 작성하시오. 수학, 물리, 컴퓨터를 멤버변수로 하고 과목들의 성적을 입력받아서 총합과 평균을 구할 것

다음의 코드를 작성한다.

Main.cpp

```
#include "Student.h"

int main(void)
{
   Student ds;
   ds.input_score();
   ds.calc_total();
   ds.average();
   ds.print_average();
```

```
    Student sj;
    sj.input_score();
    sj.calc_total();
    sj.average();
    sj.print_average();

    return 0;
}
```

Student.h

```
#ifndef __STUDENT_H__
#define __STUDENT_H__

class Student{
private:
        int math;
        int physics;
        int computer;

        int total;
        float ave;
        char credit;
public:
        void input_score(void);
        void calc_total(void);
        void average(void);
        void print_average(void);
};

#endif
```

Student.cpp

```
#include <iostream>
#include "Student.h"

using namespace std;

/* 이미 Student 라고 명시되어 있기 때문에 C 언어에서 했던 것처럼
 Student. Student-> 가 필요없다*/

void Student::input_score(void){
    cout << "Input math, physics, computer score" << endl;
    cin >> math;
    cin >> physics;
    cin >> computer;
}

void Student::calc_total(void){
    total = math + physics + computer;
}

void Student::average(void){
    ave = (float)(total / 3.0);
}

void Student::print_average(void){
    cout << "Average = " << ave << endl;
}
```

컴파일한다.

g++ Student.cpp main.cpp

13. C++의 생성자 (Constructor)

https://blog.naver.com/madplay/220057782224

생성자는 클래스 이름과 함수의 이름이 같다

소스코드 작성하고 -g 옵션 주어서 디버깅 해보자

메인 들어가자마자 s 누르면 함수이름 나온다

```
#include <iostream>

using namespace std;

class A{
        int i, j;
public:
A(void){
        cout << "생성자 호출" << endl;
        i = 10, j = 20;
}
void ShowData(void){ cout << i << ' ' << j << endl; }
};

int main(void){
        A a;
        a.ShowData();
        return 0;
}
```

다음과 같이 디버깅 옵션을 주어 컴파일 한다.

g++ -g constructor.cpp

gdb ./a.out → b main → r

14. 생성자 이용하여 자동차 클래스 만들기

자동차 클래스는 속도(속력, 방향), 색상, 기름 에 대한 정보가 나와야 한다.

Car.h

```
#ifndef __CAR_H__
#define __CAR_H__
#include <string.h>
class Car{ // 생성자를 활용하여 만들 것!
private:
        float speed;
        char color[10];
        float fuel;
public:
        Car(float s, char *c, float f)      {
                speed = s;
                strncpy(color, c, strlen(c) + 1);
                fuel = f;
        }
        void input_data(void);
        void print_car_info(void);
};
#endif
```

Car.cpp

```
#include "Car.h"
#include <iostream>
using namespace std;
void Car::input_data(void){
    cout << "Input your data(speed, color, fuel)" << endl;
    cin >> speed;
    cin >> color;
    cin >> fuel;
} // 입력받는게 의미가 없다 이미 생성자가 있으므로
```

```
void Car::print_car_info(void){

    cout << "speed = " << speed;

    cout << " color = " << color;

    cout << " fuel = " << fuel << endl;

}
```

CarMain.cpp

```
#include "Car.h"

int main(void){

    Car tesla(30, "blue", 11.1);

    tesla.print_car_info();

    return 0;

}
```

15. 소멸자

```
#include <cstring>
//#include <string.h> cstring 과 같은 기능을 한다
#include <iostream>

using namespace std;

class Academy // 생성자
{
    char *name;
    char *phone;
    public:
    Academy(char *_name, char *_phone);
    ~Academy(void); // 소멸자 앞에는 항상  ~ 가 붙는다
    void ShowData(void);
};

Academy::Academy(char *_name, char *_phone)
{
    name = new char[strlen(_name) + 1];
    strcpy(name,_name);

    phone = new char[strlen(_phone)+1];
    strcpy(phone, _phone);
}

Academy::~Academy(void)
{
    cout << "소멸자 호출" << endl;
    delete []name;
    delete []phone;
}
```

```
void Academy::ShowData(void)
{
    cout << "name: " << name << endl;
    cout << "phone: " << phone << endl;

}

int main(void)
{
    Academy aca("Bit", "02-111-2222");
    aca.ShowData();
    return 0;
}
```

http://pacs.tistory.com/entry/C-%EC%83%9D%EC%84%B1%EC%9E%90%EC%99%80-%EC%86%8C%EB%A9%B8%EC%9E%90-Constructor-Destructor-%EB%91%90%EB%B2%88%EC%A7%B8?category=367869

https://www.programiz.com/cpp-programming/library-function/cstring/strlen

***. 화면분할 명령어**

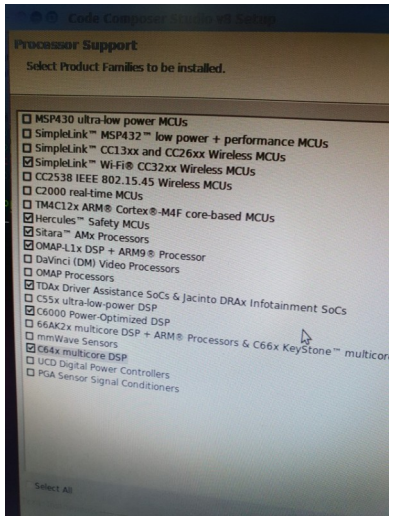
sp, vp, ^ww

http://noon.tistory.com/1353

***. DSP**

- 카페검색: AM5728 작업방법

- ccs8 설치



- ti-processor-sdk-linux-am57xx-evm-04.03.00.05 설치

***. 동적할당 관련 추가정리**

```
#include <iostream>

using namespace std;

int main()
{

	int *ptr = new int;

	*ptr =100;

	int *arr = new int[3];

	arr[0] =10;
	arr[1] =20;
	arr[2] =30;


	cout << *ptr << endl;
	cout << arr[0] << endl;

	delete ptr;
	delete[] arr;

	ptr=NULL;
	arr= NULL;
// delete 를 이용해서 메모리를 해제하면
// 해당 포인터 변수를 NULL 로 초기화 해주어야 한다.

	return 0;
}
```