|  |  |
| --- | --- |
| **Program / Project Name:** | Programming |
| **Checklist / Template Completed by:** | TBD |
| **Date Completed:** | Click here to enter a date. |

**Programming Guide**

Revision: 0.1

**Detailed Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Editor(s)** | **Description of change** |
| 0.1 | 09-JUN-2019 | Sang-Gu Kang | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 API Library 4](#_Toc11330867)

[1.1 문자열 처리 함수 4](#_Toc11330868)

[1.1.1 strcpy vs strcpy\_s 4](#_Toc11330869)

[2 수식과 입출력 4](#_Toc11330870)

[2.1 Type Cast 4](#_Toc11330871)

[2.1.1 static\_cast<type>(value) 4](#_Toc11330872)

[2.2 Type int 4](#_Toc11330873)

[2.2.1 서식 지정자 5](#_Toc11330874)

[2.2.2 min/max 6](#_Toc11330875)

[2.3 Literal 접미사 – 정수 6](#_Toc11330876)

[3 Header file 7](#_Toc11330877)

[3.1 stdint.h 7](#_Toc11330878)

# API Library

## 문자열 처리 함수

### strcpy vs strcpy\_s

가급적 strcpy\_s, sprintf\_s, strcat\_s와 같은 안전한 스트링 함수를 사용해야 한다. Visual studio에서 strcpy와 같은 함수를 사용하면 error로 처리된다. 이 함수들의 원형을 살펴보면 두 번째 인자가 size\_t numberOfElements임에 유의한다. Sizeof(buffer)가 아니라 \_countof(buffer)로 계산해야 한다.

errno\_t strcpy\_s(char \*strDest, size\_t numberOfElements, const char \*strSrc);

안전한 스트링 함수는 strDest 버퍼를 넘어서는 메모리 복사를 방지하는 함수이다. 기존의 strcpy같은 함수를 사용할 때에는, 잘못된 메모리 할당으로 어느 순간에는 문제가 없는 것처럼 돌아가다가 나중에 다른 곳에서 메모리 할당시 엉뚱하게 문제가 발생할 수 있고, 이는 디버깅 자체를 어렵게 만든다. strcpy\_s 함수는 메모리 에러를 발생시켜 주므로 메모리를 안정적으로 사용할 수 있도록 해준다.



# 수식과 입출력

## Type Cast

### static\_cast<type>(value)

형변환은 위와 같은 형식으로 엄격히 적용하는 것이 좋다. 이유는 인터넷을 참고한다.

## Type int

정수형 자료형은 크게 char, int가 있으며 앞에 부호 키워드(signed, unsigned)와 크기(short, long)를 붙여서 특성을 정의할 수 있다.



### 서식 지정자

printf 함수에서 자료형에 따라 서식 지정자를 달리 해야 한다.

* char, short, int: %d
* long: %ld
* long long: %lld
* unsigned char, unsigned short: %d
* unsigned int: %u
* unsinged long: %lu
* unsigned long long: %llu

%d: decimal  
%ld: long decimal  
%lld: long long decimal  
%u: unsigned decimal  
%lu: long unsigned decimal  
%llu: long long unsigned decimal

### min/max

int의 최소값은 -2,147,483,648로 되어 있지만, visual studio나 codeblock에서 이 값을 직접 넣어보면 waringing 혹은 error가 발생한다. 소스코드에서 정수의 최소값을 표현하려면 limits.h 헤더 파일을 사용해야 한다.



값을 계산하다가 오버플로우나 언더플로우 현상이 발생하면 의도치 않은 결과가 나올 수 있다. 따라서 프로그래밍할 때에는 정수 자료형의 크기를 항상 생각하고, 값이 범위를 넘어서지는 않는지 반드시 확인해야 한다.

## Literal 접미사 – 정수

컴파일 경고를 방지하려면 아래와 같이 뒤에 자료형 크기에 맞는 정수 리터럴 접미사를 붙여주면 된다.

unsigned int num3 = 4123456789U;

## Type float double



유효자릿수는 실수를 일정 자릿수만큼만 표현할 수 있다는 뜻이다. 만약 유효자릿수가 7이라면 0.123456789는 반올림하여 0.1234567로 표시된다. 즉, 정수 부분 1자리와 소수 부분 6자리로 7자리가 표시된다.

# Header file

## stdint.h

CPU와 OS에 따라 정수 자료형의 크기도 그때 그때 달라져서 혼란을 가져왔기 때문에 C99 표준부터 stdint.h 헤더 파일이 추가되었다.

int8\_t  
int16\_t  
int32\_t  
int64\_t  
uint8\_t  
uint16\_t  
uint32\_t  
uint64\_t

이런 자료형은 크기를 정확하게 표현해야 하는 파일 압축 및 암호화, 네트워크 프로그래밍을 할 때 특히 유용하다. stdint의 최소, 최대값은 stdint.h 헤더 파일 안에 정의되어 있으므로 limits.h 헤더 파일을 사용하지 않아도 된다.