<소프트웨어개발자를 위한 C프로그래밍>

[소프트웨어 교육과 c언어의 시작]

1. 프로그램: 컴퓨터를 실행시키기 위해 차례대로 작성된 명령어 모음
2. 프로그래밍: 작업을 정리해서 순서를 정하고 컴퓨터 특유의 명령코드로 고쳐 쓰는 작업
3. 프로그램 작성 과정
4. 문제의 정의 및 분석: 프로그램의 목적, 사용자, 입출력, 데이터 처리에 관한 요구사항, 구현상의 문제 분석, 타당성 조사 등 전과정에 대한 문서화
5. 알고리즘 설계: 계층 차트, 순서도 또는 유사 코드를 사용해서 프로그램의 논리적인 실행 순서와 내용을 구체화
6. 프로그램 코딩: 적정한 고급 언어를 선택해서 구문에 맞는 프로그램 작성
7. 프로그램 테스팅: 프로그램 디버깅(알파테스팅)과 실질적인 데이터와 실질적인 사용자에 의한 시험 실행(베타테스팅)
8. 프로그램 문서화/유지 보수: 하드웨어와 소프트웨어 요구사항, 입출력과 프로그램 파일 관리 등 전체적인 사용 설명서 작성, 프로그램의 효과적인 사용을 위한 유지 및 보수
9. 알고리즘: 어떤 주어진 문제를 논리적으로 해결하기 위해 구성된 일련의 순서화된 절차, 방법, 명령어들을 모아놓은 것
10. 알고리즘의 조건

-입력과 1개 이상의 출력

-종료되어야 함

-모든 명령이 실행 가능해야 함

알고리즘 표현 방법: 자연어, 순서도, 가상 코드, 프로그래밍 언어

1. 정렬 알고리즘: 데이터를 일정한 규칙에 따라 재배열하는 것을 의미

-오름차순 정렬(작은 수부터 큰 수 순으로 나열)

-내림차순 정렬(큰 수부터 작은 수 순으로 나열)

1)선택 정렬: 정렬되지 않은 데이터들에 대해 가장 작은(큰) 데이터를 찾아 가장 앞에 데이터와 교환해 나가는 방식

-> 가장 작은, 가장 큰 데이터를 교환한 다음 두번째로 작은, 큰 데이터를 교환 이런식으로 반복

1. 프로그램 실행 과정(절차)

원시파일 작성 ->(컴파일러) 목적 파일 생성 ->(링커) 실행 파일 생성

-원시파일: 사용자가 편집기(메모장이나 워드패드)를 이용해서 만든 파일

-컴파일러: 사용자가 만든 원시 파일을 기계어로 바꿔줌

목적 파일의 확장자는 .obj

-링커

: 소스코드와 시스템 라이브러리 파일을 연결시킴

목적 파일 여러 개를 연결시켜 실행 파일 1개를 생성

실행파일의 확장자는 .exe

[기본 문법 알아보기]

1. c프로그램의 구조

-1개 이상의 함수로 구성

-main()함수 반드시 존재

-그 외의 함수는 사용자가 임의로 작성해서 사용 가능

(1) 전처리기 지시자

-항상 #으로 시작, 한 줄에 지시자 1개만 쓸 수 있음

-헤더파일을 포함시키거나 상수 및 간단한 매크로를 정의할 때 사용

-문장의 마지막 세미콜론 x

Ex) printf()-<stdio.h>

(2) main(void) 함수

-프로그램의 시작을 알리는 함수

-c프로그램은 main() 함수 1개와 하위함수로 구성됨

(3)중괄호

-명령문들의 블록

(4)일반 명령문들

-실행 순서대로 기술해놓은 부분

-실행 순서 : 위->아래, 왼쪽->오른쪽, 경우에 따라 바꾸는 것 가능

(5)함수의 반환

-return문으로 함수의 실행 결과를 반환함

2. 문장

-문장 또는 구문이 끝날 때 세미콜론을 붙임

-세미콜론 생략하면 에러

3. 주석: 소스코드를 설명하는 부분을 주석(컴파일러는 프로그램을 실행 가능한 코드로 변환할 때 이러한 주석을 무시)

-행 단위 주석

: 코드 맨 앞에 //를 사용하면 해당 줄 주석 처리

: 코드를 작성한 뒤 맨 뒷부분을 //로 주석 처리

Printf(“hello world\n”); // 주석 처리된 문장

-블록 단위 주석

: /\*로 시작하여 \*/로 끝나며 여러 줄을 주석 처리 가능

[식별자, 변수, 상수]

1. 키워드

-고유한 의미를 갖는 예약된 단어

-예약어라고 부르기도 함

1. 식별자

-키워드 외에 프로그램에서 사용자가 필요에 따라 이름을 만들어 사용하는 단어

-식별자 작성 규칙

* 영문 대/소문자, 숫자, 밑줄(\_)을 포함한 63개의 문자로만 구성
* 첫 글자로 숫자 사용 x
* 대소문자 구별하고 키워드 사용 x

1. 상수 : 프로그램의 실행 시작부터 끝날 때까지 값이 변하지 않는 자료
2. 리터럴 상수: 문자는 ‘ ‘로 표시, 문자열은 “ “로 표시
3. 기호 상수, 정의된 상수, 매크로 상수

: #define PI 3.141592  
전처리문인 #define을 이용해 상수명과 값을 지정

1. 메모리 상수

const double PI = 3.141592

* 변수일 때는 변경 가능하지만 상수일 때는 값 변경 x

1. 변수 : 프로그램에서 임시로 자료 값을 저장할 수 있는 저장 장소의 이름

변수에 값을 저장할 수 있고 이 값은 계속 변경

먼저 변수 선언을 해야 사용 가능

-변수 선언의 예

1)자료형: 사용하는 변수 종류 예) int

2)변수명: 실제로 정수값이 저장되는 곳의 이름 예) age

-변수의 초기화: 선언된 변수에 처음으로 값을 저장하는 것

-초기값이 저장되기 전 변수에는 알 수 없는 무의미한 값 저장

-변수 동시 선언: 동일한 자료형의 변수를 여러 개 사용해야 할 경우, 동시에 선언

예) int a = 10, b = 20;

[정수형 자료형]

1. 자료형: 프로그램에서 선언된 변수들이 어떻게 저장되고 처리되어야 할지, 컴파일러에 알려줌
2. 정수형

-소수점이 없는 숫자

-양수, 음수에 따라 부호 사용 가능

-short, int, long

-(생략 가능)signed(부호 o), unsigned(부호 x)

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 오버플로우: 데이터의 허용 범위를 넘는 값을 변수에 저장할 때, 사용자가 의도한 값이 아닌 전혀 다른 값이 저장되는 것텍스트, 스크린샷, 원, 폰트이(가) 표시된 사진

   자동 생성된 설명
2. 언더플로우

[실수형 자료형]

1. 실수형: 소수점이나 지수가 있는 수

-float, double, long double

Float a = 3.14f;

Double a = 5.6847;

Long double a = 5.69424L

1. 출력할 때 서식문자(변환기호): printf()문

-float : %f

-double : %f, %lf

-long double : %f %lf %llf

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 출력 폭, 방향, 소수점 자리수

-값은 기본적으로 우측 정렬

-(+)는 출력시 부호와 값이 함께 출력

printf(“%+6d”,20) -> [\_ \_ \_ + 2 0]

-(-)는 출력되는 값을 왼쪽 정렬

printf(“%-6d”,20) -> [2 0 \_ \_ \_ \_]

-소수점 자리수 지정

printf(“%6.1f”,5.06); -> 1은 소수점의 자리수를 1자리로 한다는 의미

\*소수점 자리에 맞게 숫자는 반올림함

[문자형]

1. char : 문자형을 다루는 자료형

-문자 1개를 작은따옴표(‘ ‘) 사이에 넣어 사용

-8비트로 처리됨

-내부적인 문자 코드(아스키 코드)에 상응하는 숫자로 바뀌어 기억됨

-0~127 사이의 부호 없는 정수에 문자를 정의

2. c언어에서 다루는 문자

-특수문자

\a: 벨소리를 냄

\b: 왼쪽으로 한 문자를 지움

\f: 프린터에서 한 페이지를 이동

\n: 새로운 줄로 바꿈

\r: 커서를 1열로 이동

\t: 한 탭을 이동

\’, \”: ‘, “를 출력

\0: null 문자를 처리함

1. 아스키 코드: 숫자와 문자를 연결시켜주는 프로토콜

Dec: 정수

Hex: 16진수

‘A’ =65

‘a’= 97

[형변환]

1. 형변환: 하나의 자료형을 다른 자료형으로 변환시키는 것
2. 묵시적 형변환:

-컴파일러가 자동으로 변환

-데이터의 값을 잃지 않는 쪽으로 형변환이 이루어짐

Char -> short -> int -> unsigned -> long -> float -> double

-정수를 실수로 형변환하는 경우

-실수를 정수로 형변환하는 경우

(2) 명시적 형변환

- 사용자가 강제로 자료형 변환시키는 것

- 형변환 연산자(캐스팅 연산자)를 사용

- 연산자 앞에 형변환 연산자()를 붙이고 ()안에 변환시키려는 자료형

* 정수형/정수형 = 정수형
* 정수형/실수형 = 실수형
* 실수형/정수형 = 실수형
* 실수형/실수형 = 실수형

[표준 입출력]

1. 입출력 함수

-파일 입출력 함수: 특정한 파일을 통해 자료를 입출력

-저급 입출력 함수: 운영체제 내의 시스템 호출을 통해 자료를 입출력

1. 표준 입출력 함수: 키보드와 모니터를 통해 자료를 입출력

-형식화된 입출력 함수: scanf(), printf()

-문자 입출력 함수: getchar(), putchar()

-문자열 입출력 함수: gets(), puts()

(1) printf() 함수

: 표준 출력 장치인 모니터로 자료를 출력하는 함수

1. 기본 형식

Printf(“문자열”);

Printf(“제어 문자열”, 변수나 표현식);

1. 제어 문자열

* %d : 부호 있는 10진 정수
* %f : 실수
* %c : 문자 1개
* %s : 문자열
* %o: 부호 없는 8진 정수
* %x : 부호 없는 16진 정수
* %u : 부호 없는 10진 정수
* %lu : long형 부호 없는 10진 정수

1. scanf() 함수

: 2개 이상 자료를 입력할 때 쉼표를 넣어 변환기호와 변수를 순서대로 대응

\*\* 오류 \*\*

#pragma warning(disable:4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

1. 사용법

-정수형, 실수형 값 입력

Scanf\_s(“변환기호”, &변수명)

-문자, 문자열 값 입력

Scanf\_s(“변환기호”, &변수명, sizeof(변수명 or 자료형));

-문자, 문자열 값을 2개 이상 입력

Scanf\_s(“변환기호1 변환기호2”, &변수명1, sizeof(변수명1), &변수명2, sizeof(변수명2)):

1. 버퍼형 표준 입출력 함수: scanf\_s()

: 데이터를 입력한 다음 엔터키를 눌러야 값이 전달되는 함수

버퍼 메모리를 사용하여 키보드로 입력되는 값을 버퍼 메모리에 보관하였다가 엔터키를 누르면 데이터를 전달

1. Scanf\_s()함수 연속해서 선언하기

-버퍼 메모리에 엔터키값이 남아있어 다음 scanf\_s() 함수 사용에 지장을 초래

-getchar()는 문자 1개를 입력받는 함수지만, 입력한 순서대로 계속 버퍼에 저장했다가 문자 1개씩 가져오고 난 다음 버퍼를 하나씩 비움.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

\*\* scanf()에서는 float(%f), double(%lf) 무조건 구분해야 함 \*\*

1. 탈출 기법

텍스트, 번호, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 문자 입출력 함수: 문자 1개를 키보드로 입력받아 출력하는 함수

-getchar() 함수 : 문자 입력 함수

-putchar() 함수: 문자 출력 함수

int getchar() : 문자 1개를 입력받으며 리턴값은 입력받은 문자의 아스키 코드값임.

int putchar(char 문자): 문자 1개를 출력하며 리턴값은 출력하려는 문자에 해당하는 아스키 코드값임.

1. 문자열 입출력 함수: 문자 여러 개를 한번에 입력받고 출력하는 함수

-gets() 함수: 공백 문자를 포함한 문자열을 입력

-puts() 함수: 표준출력 장치로 주어진 문자열을 출력할

Char\*gets(char\*str) : 키보드에서 한 행을 입력받음. 리턴값은 읽은 문자열

Int puts(char\*str) : 모니터에 한 행을 출력. 리턴값은 양수

get\_s(버퍼,버퍼크기)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

\*\*배열명 사용할 때는 scanf에서 주소 & 안 적어도 됨.\*\*

[연산자]

1. 연산자의 종류

-연산자: 산술 연산자인 +, -, \*, / 와 같이 이미 정의된 연산을 수행하는 기호

-피연산자: 연산에 참여하는 변수나 값

|  |  |
| --- | --- |
| 산술 연산자 | +, -, \*, / |
| 관계 연산자 | >, <, >=, <=, ==, !=(같지 않다) |
| 증감 연산자 | ++, -- |
| 논리 연산자 | &&(논리), ||(논리합), !(부정) |
| 조건 연산자 | ?: |
| 비트 논리 연산자 | &, |, ^, ~ |
| 비트 이동 연산자 | <<, >> |
| 대입 연산자 | = |

1. 대입 연산자(=)

lvalue = rvalue;

lvalue: 대입 연산자를 기준으로 왼쪽에 있는 피연산자(오로지 변수만 올 수 있음)

lvalue: 대입 연산자를 기준으로 오른쪽에 있는 피연산자(값, 변수, 수식이 올 수 있음)

1. 산술 연산자( +, -, \*, /)
2. 단항 연산자: 피연산자를 1개만 사용하는 산술 연산자
3. 이항 연산자: 피연산자를 2개 사용하는 산술 연산자 ex) x+y

종류: 사칙연산과 나머지 연산(%)

-나눗셈 연산자: 몫만 저장

-나머지 연산자: 나머지만 저장(\*\*only 정수로만 연산 가능\*\*)

1. 증감 연산자

++a: 변수값을 수식에 먼저 적용시킨 후 최종 변수값을 1만큼 증가

a++: 변수값을 1만큼 먼저 증가시킨 후 최종 변수값을 수식에 적용

--a: 변수값을 수식에 먼저 적용시킨 후 최종 변수값을 1만큼 감소

a--: 변수값을 1만큼 먼저 감소시킨 후 최종 변수값을 수식에 적용

1. 관계 연산자(>, <, >=, <=, ==, !=(같지 않다))

-관계가 성립되면 참(1)

-관계가 성립되지 않으면 거짓(0)

1. 논리 연산자(&&(논리), ||(논리합), !(부정))

-&&: 거짓이 하나라도 있으면 거짓

-||: 참이 하나라도 있으면 참

-!: 참->거짓, 거짓->참

\*\* 0이외의 모든 숫자는 참\*\*

-논리 연산자의 우선순위

(논리부정 > 논리곱 > 논리합)

1. 비트 연산자(&, |, ^, ~, <<, >>)

&(비트곱): and 연산하여 대입

|(비트합): OR 연산하여 대입

^(배타적 논리합): XOR 연산하여 대입(비트가 다를 때 참, 같은 때 거짓)

~(비트 반전): 비트를 반전

<<(왼쪽으로 이동)

>>(오른쪽으로 이동)

1. 오른쪽 shift 연산

a >> b : a를 b만큼 오른쪽으로 비트를 이동시켜 값을 감소

a >> b 연산의 값: a를 2^b로 나눈값

1. 왼쪽 shift 연산

a << b : a를 b만큼 왼쪽으로 비트를 이동시켜 값을 증가

a << b 연산의 값: a에 2^b로 곱한 값

1. 축약 연산자

: 연산 2개를 동시에 수행하여 값을 할당하는 연산자

Ex) +=, -=

1. Sizeof 연산자(변수나 자료형의 크기를 알고 싶을 때 사용)

\*변수는 괄호 생략 가능\*

1. 연산자 우선순위

단항 > 산술 > 이동 > 관계 > 비트 > 논리 > 조건 > 대입

()안의 내용이 최우선순위

단항 연산자가 이항 연산자보다 먼저 처리됨

대개 좌측에서 우측으로 실행

[선택문]

1. 제어문
2. 선택문

if

if ~ else

if ~ else ~ if

다중 if문

Switch ~ case

1. If 문
2. If 문 형식
3. If( 조건식 ) 문장;
4. If( 조건식 )

문장1;

Else

문장2;

1. If( 조건식 1 )

문장1;

Else if( 조건식 2 )

문장2;

Else if( 조건식 3 )

문장3;

1. If( 조건식 )

If( 조건식)

1. If ~ else 문
2. 괄호 사용 방법

:문장이 여러 개인 경우 무조건 중괄호 해야 함

If( 조건식 ) {

문장1;

문장2;

}

1. If ~ else ~ if 문

: 대등한 조건 1개가 아니라 여러 개 있을 경우에 사용

1. 다중 if 문

: if 문 안에서 또다른 조건식을 검사하고자 할 때 사용

1. Switch ~ case 문

: 조건식을 먼저 평가한 뒤, 그 식의 값이 case 상수와 일치하는 쪽으로 분기하여 해당 명령문을 수행

1. Switch문 형식

Switch(조건식(값))

{

Case 상수값 1:

명령문 블록1

Break;

Case 상수값 2:

명령문 블록2

Break;

.

.

.

Default :

명령문 블록 n

Break;(생략 가능)

]

\*\*조건문에는 반드시 수식이나 값을 넣어야 함)

1. 조건 연산자

: 유일하게 피연산자가 3개인 삼항 연산자

기능이 선택문과 같음

1. 작성 형식

(문장1) ?: (문장2 ): (문장3);

문장 1의 내용이 참이면 문장 2 실행

거짓이면 문장 3 실행