|  |  |  |  |
| --- | --- | --- | --- |
| 프로그래밍 실습 2 |  |  |  |
|  | | | |
| Report 4 | | | |
| Ch. 10 - 함수 기초 | | | |
|  | | | |
|  |  | **학번** | 2018212236 |
|  |  | **학부** | 전자정보통신공학 |
|  |  | **이름** | 김동주 |
|  |  |  |  |
|  |  | **제출일자** | 2018-10-07 |
|  |  | **담당교수** | 반상우 |
|  |  |  |  |
|  |  | **작업환경** | Visual Studio Code |
|  |  | **컴파일러** | MinGW GCC |
|  |  |  |  |

# 

# 필수 문제

## 1. p.488-01

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 인치(inch)를 센티미터(cm)로 바꾸는 함수를 작성하여, 표준입력으로 받은 인치를 센티미터로 출력하는 프로그램을 작성하시오. | | |
| **코드** | #include <stdio.h>  float inchToCm(float);  int main(void)  {  float inch;  scanf("%f", &inch);  printf("%f\n", inchToCm(inch));  return 0;  }  float inchToCm(float inch) {  return inch \* 2.54F;  } | | |
| **실행결과** |  | | |
| **고찰** | main함수 안에 문제에서 요구하는 기능을 전부 구현하는 것 보다, 필요한 기능에 따라 함수로 분리하여 구현하는 방식이 훨씬 깔끔하다고 느꼈다. | | |

## 2. p.488-02

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 두 개의 임의 정수 m, n을 입력 받아 다음 함수를 작성하여 m^n의 결과를 출력하는 프로그램을 작성하시오.   * 함수 intpow(int m, int n)의 결과는 m^n | | |
| **코드** | #include <stdio.h>  int intpow(int, int);  int main(void) {  int m, n;  scanf("%d %d", &m, &n);    printf("%d\n", intpow(m, n));  return 0;  }  int intpow(int m, int n) {  int res = 1;  for (int i = 1; i <= n; i++)  res \*= m;    return res;  } | | |
| **실행결과** |  | | |
| **고찰** | 1번 문제에서는 inchToCm 함수   |  | | --- | | float inchToCm(float inch) {  return inch \* 2.54F;  } |   의 역할이 단순 곱셈 1회 이여서 함수의 필요성이 크게 와닿지 않았는데, 2번 문제에서의 intpow함수는 반복문 + 곱셈 조합으로 주석을 따로 적어주지 않는 한   |  | | --- | | int res = 1;  for (int i = 2; i <= n; i++)  res \*= m; |   위의 코드의 역할이 무엇인지 한눈에 보고 맞추기 어려울 것이다.  하지만 함수를 사용하여 코드를 분리하니 코드의 전체적인 구조가 간단 명료해졌고, 함수의 이름만 보고도 무슨 역할의 코드인지 알 수 있어 편리하다고 느꼈다. | | |

## 3. p.488-03

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 다음 식과 내용을 참고로 섭씨온도(C)를 화씨온도(F)로 변환하는 함수를 작성하여 그 결과를 출력하는 프로그램을 작성하시오.   * 섭씨온도(C)와 화씨온도(F)와의 관계식 : C = (5/9)(F-32) * 섭씨온도가 0부터 100까지 0.5씩 증가하도록 하며, 이 때의 화씨온도를 구하여 출력, 단 온도는 모두 소수점 2자리까지 출력 | | |
| **코드** | #include <stdio.h>  float CtoF(float);  int main(void) {  float cel;  for (cel = 0; cel <= 100; cel += .5F)  printf("%.2fC = %.2fF\n", cel, CtoF(cel));  }  float CtoF(float \_cel) {  return \_cel \* (9.0F / 5.0F) + 32;  } | | |
| **실행결과** |  | | |
| **고찰** | 처음에 값이 다음과 같이 출력이 되었다.   |  | | --- | | …  95.00C = 127.00F  95.50C = 127.50F  96.00C = 128.00F  96.50C = 128.50F  97.00C = 129.00F  97.50C = 129.50F  98.00C = 130.00F  98.50C = 130.50F  99.00C = 131.00F  99.50C = 131.50F  100.00C = 132.00F |   F을 C로 변환하는 식에 9/5를 곱하는 계산이 있음에도 저런식이 나와서 그 이유를 찾아보다가   |  | | --- | | return \_cel \* **(9 / 5)** + 32; |   위 식에서 9와 5가 모두 int형이고 괄호로 묶여있어, 이 들이 먼저 계산되어지고, 그 결과 \_cel \* (1) + 32 가 반환되었던 것이다.  따라서 9와 5를 int형이 아닌 float형(9.0F, 5.0F)로 변환시켜주었고, 값은 정상적으로 출력되었다. | | |

## 4. p.488-04

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 세 개의 임의 정수를 입력 받아 가장 큰 수를 출력하는 프로그램의 함수를 구현하여 그 결과를 알아보는 프로그램을 작성하시오. | | |
| **코드** | #include <stdio.h>  int int3\_getMax(int, int, int);  int main(void) {  int a, b, c;    scanf("%d %d %d", &a, &b, &c);  printf("Max : %d\n", int3\_getMax(a, b, c));  return 0;  }  int int3\_getMax(int a, int b, int c) {  int max = a;  if (b > max)  max = b;  if (c > max)  max = c;    return max;  } | | |
| **실행결과** |  | | |
| **고찰** | 함수를 만드는 이유는 무엇일까에 대하여 생각해보았다.  함수를 만드는 이유는 아마도 같은 로직의 코드를 재사용 할 때 편의를 주기 위해서라고 생각한다. 또한 부수적으로 따라오는 이점으로 코드가 간결해진다는 것도 있다.  이러한 이점을 살리는 데에 있어, 대규모 프로젝트같은 여러 개발자가 함께 작업하는 코드에서, 내가 만든 함수를 타인이, 타인이 만든 함수를 내가 쓰게 된다면 어떨까.  유명 라이브러리 같이 이미 널리 쓰이고, 사용법이 잘 정리되어있는 함수들은 우리가 사용하는데 큰 문제가 없지만, 사용자 함수는 얘기가 다를 것이다.  가령, ‘a’라는 함수가 있다고 하면, a함수를 만든 사람이 와서 설명해주지 않는 이상, a(1), a(void), … 와 같이 어떤 자료형의 인자를 얼마나 주어야 하는지, 그 함수의 기능은 무엇인지, 주의사항은 있는지 알 수 없을 것이다.  그렇기에 함수에 관한 정보를 한 눈에 알기 쉽게 하려면 그만큼 함수명을 신중하게 지어야 겠다고 생각하였다.  readFileSync()라는 함수가 있으면   * readFile... : 파일을 읽어오는 함수 * ...Sync : 함수의 동작을 동기적으로 처리   라는 식으로 기능을 파악하고,   * 파일을 읽어오는 함수 -> 파일의 경로가 필요할 것.   과 같은 유추를 통해 바로 아래와 같이 적용하여 사용할 수 있을 것이다.  readFileSync(“C:\readme.txt”);  요약 : **함수명을 신중하게** 지어야 겠다고 느꼈다. | | |

# 선택 문제

## 1. p.489-14

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 다음과 같이 일차원 배열을 복사하는 함수를 작성하여 결과를 알아보는 프로그램을 작성하시오.   * void copyarray(int from[], int to[], int n /\* 배열 원소 수 \*/) * 배열 from의 첫 번째 원소부터 (n-1)번째 원소까지 같은 순서대로 배열 to로 값을 복사하는 함수 | | |
| **코드** | #include <stdio.h>  void copyArray(int from[], int to[], int n);  void printIntArray(int array[], int length);  int main(void) {  int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };  int b[10] = { 0 };  printf("Array 'a' : ");  printIntArray(a, 10);    printf("Array 'b' : ");  printIntArray(b, 10);  printf("Copy 'b' with 'a'\n");  copyArray(a, b, 10);    printf("Array 'b' : ");  printIntArray(b, 10);  return 0;  }  void copyArray(int from[], int to[], int n) {  for (int i = 0; i < n; i++)  to[i] = from[i];  }  void printIntArray(int array[], int length) {  for (int i = 0; i < length; i++)  printf("%d ", array[i]);  puts("");  } | | |
| **실행결과** |  | | |
| **고찰** | 함수 copyArray의 리턴형이 void여서 ‘호잉?’ 하고 보았다.  왠지 copy 작업을 하고나면, 무언가(copy된 새로운 배열이라던가, ...)를 반환할 것이라 당연하게 생각하였다.  그런데 그럴 필요가 없는 것이 당연한게, C언어의 배열은 어떻게 보면 포인터의 성질을 가지고 있어서, 그 자체가 메모리의 주소값이기에, 입력받은 배열만 가지고도 scope의 영향을 받지 않고 수정이 가능했던 것이다.  호옷!  이런 맛에 C언어를 사용하는 것일까? 라고 생각했다. | | |

## 2. p.489-15

|  |  |  |  |
| --- | --- | --- | --- |
| **내용** | 다음과 같이 일차원 배열의 동등함을 검사하는 함수를 작성하여 결과를 알아보는 프로그램을 작성하시오.   * int isequalarray(int a[], int b[], int n /\* 배열 원소 수 \*/) * 배열 a와 b의 배열크기가 모두 n이며 순차적으로 원소 값이 모두 같으면 1을 반환, 아니면 0을 반환하는 함수 | | |
| **코드** | #include <stdio.h>  int isEqualArray(int a[], int b[], int n);  void printIntArray(int array[], int length);  int main(void) {  int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };  int b[10] = { 0 };  int c[8] = { 2, 4, 8, 16, 32, 64, 128, 256 };  int d[8] = { 2, 4, 8, 16, 32, 64, 128, 256 };  printf("Array 'a' : ");  printIntArray(a, 10);    printf("Array 'b' : ");  printIntArray(b, 10);  printf("Array 'c' : ");  printIntArray(c, 8);    printf("Array 'd' : ");  printIntArray(d, 8);    printf("is 'a' equals 'b' : ");  if (isEqualArray(a, b, 10))  puts("true");  else  puts("false");  printf("is 'c' equals 'd' : ");  if (isEqualArray(c, d, 8))  puts("true");  else  puts("false");  return 0;  }  int isEqualArray(int a[], int b[], int n) {  for (int i = 0; i < n; i++) {  if (a[i] != b[i])  return 0;  }  return 1;  }  void printIntArray(int array[], int length) {  for (int i = 0; i < length; i++)  printf("%d ", array[i]);  puts("");  } | | |
| **실행결과** |  | | |
| **고찰** | 이 문제에서 사용된 isequalarray함수는 입력받은 두 배열이 가진 값이 같은지 ‘참/거짓’ 여부를 반환해야 하는 함수이다.  타 언어를 사용하다가 C언어를 사용하면서 자주 당황하게 되는 것이 C언어에는 **boolean** 자료형이 기본으로 제공되지 않는다는 점이었다.  그렇기에 boolean이 있으면 좋겠다라고 생각하는 문제를 풀 때에는, 아래와 같이 전처리기를 사용하여 true와 false를 만들어 사용하였다.   |  | | --- | | #define TRUE 1  #define FALSE 0  int isEqualArray(int a[], int b[], int n) {  for (int i = 0; i < n; i++) {  if (a[i] != b[i])  return FALSE;  }  return TRUE;  } |   그러다 문득 다른사람들은 어떻게 이를 해결하는지, 기발한 방법이 있지는 않을지 찾아보던 중, <stdbool.h>이라는 헤더파일을 알게되었다.  이 친절하고 좋은 헤더파일은 bool이라는 자료형과 함께 true와 false를 제공한다!!  이를 이용하여 위 문제의 isEqualArray함수를 작성해보았다.   |  | | --- | | bool isEqualArray(int a[], int b[], int n) {  for (int i = 0; i < n; i++) {  if (a[i] != b[i])  return false;  }  return true;  } |   확실히 함수의 반환형이 bool이라고 제시되어 있으니, true혹은 false가 반환되는 함수이겠구나라고 파악하기 쉽겠다는 생각이 들었다.  함수의 반환형이 int형이면, 아무리 0이 false, 그 외가 true의 역할을 한다 해도, 한번에 알아보기 어려울 것이다.  char은 int와 비슷하게 숫자로서 사용이 가능하고, int는 char와 같이 %c를 출력하는데 사용할 수 있음에도 불구하고, char은 문자를 사용하기 위해서, int는 정수를 사용하기 위해서만 사용하는게, 가독성을 향상시키는데 일조하는 것 처럼  **<stdbool.h>**을 사용한다면 코드가 조금은 **더 읽기 편하고 재밌어**지지 않을까라고 생각하였다. | | |