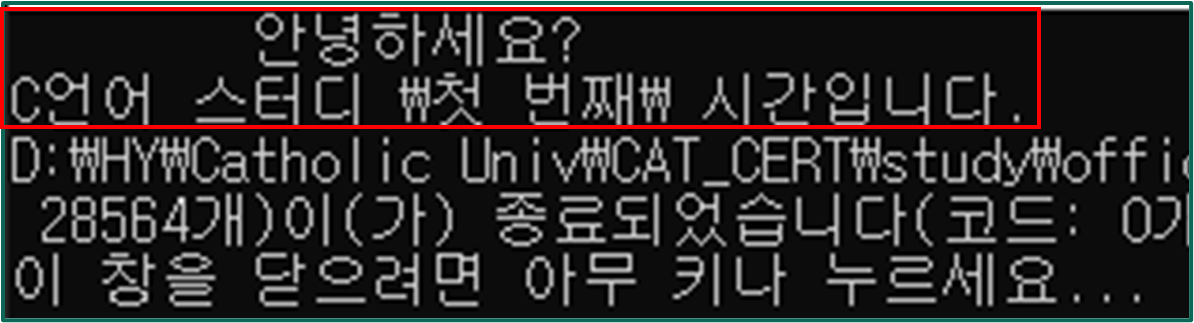
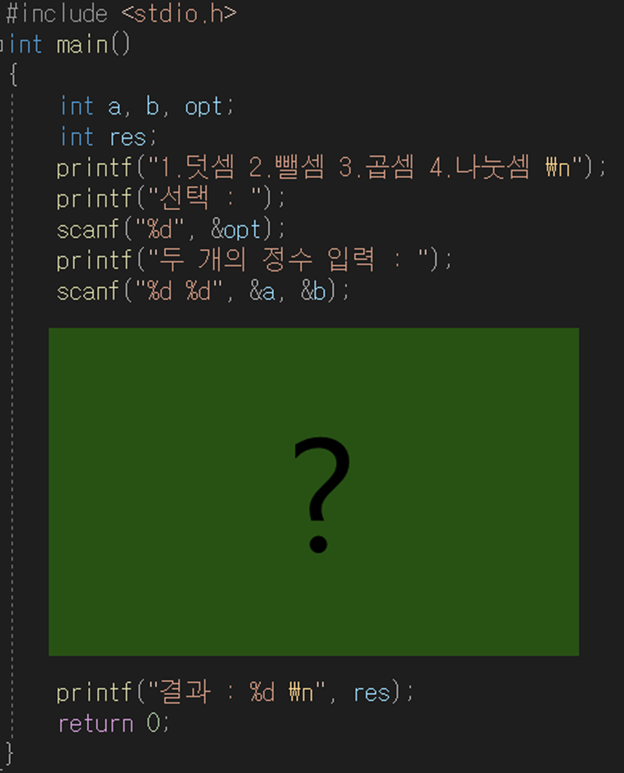
**실습 1**



|  |
| --- |
| #include <stdio.h>  int main() {  printf("\t안녕하세요? \n C언어 스터디 \\첫 번째\\ 시간입니다.");  return 0;  } |

**IF문 실습**



|  |
| --- |
| #include <stdio.h>    int main()  {  int a, b, opt;  int res;  printf("1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 \n");  printf("선택 : ");  scanf("%d", &opt);  printf("두 개의 정수 입력 : ");  scanf("%d %d", &a, &b);    if (opt == 1)  res = (a + b);  else if (opt == 2)  res = (a - b);  else if (opt == 3)  res =(a \* b);  else if (opt == 4)  res = (a/b);  printf("결과 : %d \n", res);  return 0;  } |

**IF문 실습 2**

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| #include<stdio.h>  int main()  {  int a;  printf("숫자 입력 : ");  scanf("%d", &a);  if (a % 3!=0 && a % 5 != 0)  printf("%d 는 3과 5의 배수가 아닙니다. ", a);  else if (a % 3==0 && a % 5 == 0)  printf("%d 는 3과 5의 배수입니다. ", a);  return 0;    } |

**IF문 실습 3**



|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  int main(void) {  int a;  printf("연도 입력 : ");  scanf("%d", &a);  if (a % 4 == 0 && a % 100 != 0)  printf("%d년은 윤년입니다.", a);  else if (a % 400 == 0)  printf("%d년은 윤년입니다.", a);  return 0;  } |

**switch문 실습**

달을 입력 받은 뒤 몇 일 까지 있는지 출력하라

|  |
| --- |
| #include <stdio.h>    int main()  {  int a;  printf("Month Input : ");  scanf("%d", &a);  switch(a) {  case 1: case 3: case 5: case 7: case 9: case 11 :  printf("31일까지 있습니다.");  break;  case 4: case 6: case 8: case 10: case 12 :  printf("30일까지 있습니다.");  break;  case 2:  printf("28일까지 있습니다.");  break;      }  return 0;  } |

**삼항연산자 실습**

|  |
| --- |
| #include <stdio.h>    int main()  {  int a,b;  scanf("%d %d", &a, &b);    printf("%d", (a>b) ? b : a);  } |

**실습 과제 1) 합**

|  |
| --- |
| #include <stdio.h>    int main()  {  int n;  scanf("%d", &n);    printf("%d", n\*(n+1)/2);    return 0;  } |

**실습 과제2) 과일 옮기기**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  int main()  {  int a, b, c, d;  int fruit = 0;  scanf("%d %d", &a, &b);  scanf("%d %d", &c, &d);  if (a + d < b + c)  fruit = a + d;  else  fruit = b + c;  printf("%d", fruit);  return 0;  } |

**실습 과제 3) 두 번째로 큰 수 구하기**

|  |
| --- |
| #include <stdio.h>    int main()  {  int a, b, c;  scanf("%d %d %d", &a, &b, &c);    if (a > b && b > c)  printf("%d", b);    else if (c > b && b > a)  printf("%d", b);    else if (b > c && c > a)  printf("%d", c);    else if (a > c && c > b)  printf("%d", c);    else if (b > a && a > c)  printf("%d", a);    else if (c > a && a > b)  printf("%d", a);      return 0;  } |

**실습 과제4) ABCD 외우기**

|  |
| --- |
| #include <stdio.h>    int main()  {  char a, b, c;  scanf("%c %c %c", &a, &b, &c);  if (a >= 'a' && a<= 'z')  a = a - ('a'-'A');    if (b >= 'a' && a <= 'z')  b = b - ('a' - 'A');    if (c >= 'a' && a <= 'z')  c = c - ('a' - 'A');      printf("%c", (a<b && b<c) ? 'O' : 'X');    return 0;  } |

**용어 정리**

**#1 컴파일러에 대해 조사하기 (컴파일언어와 인터프리터언어(≠스크립트언어)의 차이점 조사)**

컴파일러: 고급언어로 쓰인 프로그램을 그와 의미적으로 동등하며 컴퓨터에서 즉시 실행될 수 있는 형태의 목적 프로그램으로 바꾸어 주는 번역 프로그램.

* 컴파일하는 과정이 복잡하고 그 시간이 많이 걸리지만 한 번 컴파일하면 그대로 사용이 가능
* 컴파일하는 단계 별로 주기억장치 안에 들어올 수 있기 때문에 기억장소의 크기는 조정 될 수 있음
* 반복하는 경우에 기계코드가 반복을 하기 때문에 처리시간이 빠르고 실행시간 면에서 효율적임.
* 한 줄로 명령어를 입력했을 경우 프로그램이 때로 몇 백 개의 기계어로 변역되기 떄문에 큰 기억용량을 요구함.

|  |  |
| --- | --- |
| 컴파일러 언어 | 인터프리터 언어 |
| 소스 코드를 전체로 한 번 훑고 컴퓨터 프로세서가 실행 할 수 있도록 바로 기계어로 변환 | 고레벨 언어를 중간 코드로 변환하고 이를 각 행마다 실행 ( 중간 코드는 다른 프로그램에 의해 실행) |
| 전체 소스코드를 변환 한 뒤 에러를 보고 | 각 행마다 실행하는 도중 에러가 보고 되면 이후 코드는 살펴보지 않음 (보안적인 관점에서 도움) |
| C, C++ | R, Python, Ruby |

**#2 디버그모드와 릴리즈모드의 차이 조사**

디버깅 정보를 실행코드 안에 넣는 것의 유무 차이

|  |  |
| --- | --- |
| 디버깅 모드 | 릴리즈 모드 |
| 실행파일에 디버깅 정보를 삽입하여 언제든 디버깅을 할 수 있음 | 같은 문자열 상수라도 서로 다른 공간에 할당 |
| 디버깅 정보가 들어가 있기에 실행 파일 상태를 확인할 수 있음 | 디버깅정보를 삽입 X, 코드를 최적화하여 실행 파일 크기를 최대한 줄임 |
| 정보 실행시 계속 확인하기에 속도가 느려짐 | 속도나 크기면에서 유리함 |

* 디버깅 모드로 컴파일하게 되면 실행상태에서 추적할 수 있는 정보가 실행파일 안에 들어가게되므로 용량이 커짐
* 릴리즈모드의 경우 디버깅 정보없이 순수한 소스코드 자체의 기능만 컴파일되어 실행파일로 만들어짐.

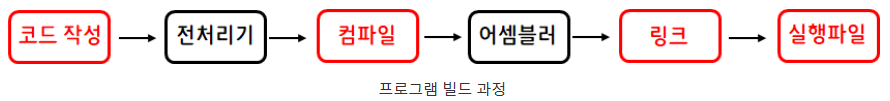
**#3 리틀엔디안 vs 빅엔디안 조사 (0x41424344가 빅엔디안일 때 리틀엔디안으로는?)**

엔디안: 컴퓨터의 메모리와 같은 1차원의 공간에 여러 개의 연속된 대상을 배열하는 방법

|  |  |
| --- | --- |
| 리틀엔디안 | 빅엔디안 |
| 최하위 바이트부터 차례로 저장 | 최상위 바이트부터 차례로 저장 |
| 작은 단위가 앞에 나옴 | 큰 단위가 앞에 나옴 |
| 메모리에 저장된 값의 하위 바이트들만 사용할 때 별도의 계산이 필요 X | 디버그를 편하게 해줌 (사람과 숫자를 읽고 쓰는 방법이 같음) |

0x44 43 42 41

**#4 .c파일이 실행파일이 되기까지의 과정 조사**



1. 코드 작성

: 코드 작성

1. 전처리기

: 컴파일에 들어가기 전에 작성한 소스 코드가 정상 작동하도록 # (전처리 지시자)의 지시에 맞게 컴파일 전에 단순 치환하는 단계

전처리 과정 :　＃include <stdlib.h>를 파일 안의 내용으로 바꿔주는 과정

3. 컴파일

: 사용자가 작성한 고수준 언어를 어셈블리어로 한 단계 번역하는 단계

4. 어셈블러

: 번역된 어셈블리어를 기계어로 최종 번역하는 단계

* 소스코드에 오류가 없담녀 컴파일러는 각 문장들을 기계어로 변환

5. 링크

오브젝트 파일들을 라이브러리(프로그래머들이 많이 사용하는 기능을 미리 작성해 놓은 것, 컴파일러에 내장되어 있음)와 하나로 묶는 단계

6. 실행 파일

: 앞 과정을 거치게 되면 실행 파일이 생성됨.

**#5 기계어와 어셈블리어에 대해 조사**

|  |  |
| --- | --- |
| 기계어 | 어셈블리어 |
| 컴퓨터의 CPU가 명령을 처리할 때 사용하는 언어 | 컴퓨터 프로그래밍의 저급 언어 |
| 2진법으로 구성되어 있음 | 0과 1을 사람이 읽을 수 있는 언어로 변환 (기호언어) |
| 프로그램을 나타내는 가장 낮은 단계의 개념 | 저급 언어 중에서도 밑바닥의 언어 |
| 대부분 어셈블리어를 거쳐 짜여짐 | 바로 명령을 내리기에 실행 속도가 빠름 |
| 컴퓨턱 알아들을 수 있는 유일한 프로그램이 언어 | 최소한의 명령을 내리기에 프로그램의 크기가 작음 |