



C 언어 프로그래밍

Part 02. C 언어 기본 학습

Chapter 06. 반복문

목차

1. 반복문이란?
2. for문
3. while문
4. do~while문
5. 반복문 고급

[실전예제] 100 이하의 소수 출력하기

학습목표

- 반복문의 특징을 이해하고 종류를 알아본다.
- for문의 개념과 실행 흐름을 이해한다.
- for문과 while문을 비교하여 적용할 수 있다.
- while문과 do~while문의 차이점을 이해한다.
- 다중 반복문을 이해하고 break와 continue의 쓰임새를 알아본다.

01

반복문이란?

01. 반복문이란?

I. 반복문

- 반복문의 개념

- 제어문의 일종으로 특정한 코드를 반복적으로 실행시키는 명령문

- 반복문의 특징

- 0회 이상 반복
- 반복하는 횟수를 설정하는 기능은 없음
- 특정 횟수만큼 반복 실행되도록 할 수 있음

01. 반복문이란?

I. 반복문

- 반복문의 종류

- for문 : 제어 변수를 조건식에 사용하여 특정 횟수만큼 반복 실행
- while문, do~while문 : 특정 조건이 만족할 때까지 반복 실행



그림 6-1 for문과 while문의 비교

01. 반복문이란?

확인문제1

C 언어에서 어떤 코드를 특정 횟수만 반복하는 방법을 설명하시오.

01. 반복문이란?

LAB 6-1

입력한 횟수만큼 반복하기

자기 자신을 호출하는 함수 `void Print(int count)`를 이용하여 입력받은 반복 횟수만큼 "난생처음"을 출력하는 프로그램을 작성해봅시다. 이때 `count`가 0보다 클 경우 출력한 뒤에 `count-1`을 인자로 자기 자신 `Print`를 다시 호출하는 방법을 사용합니다.

반복 횟수를 입력하세요!

3

난생처음 난생처음 난생처음

반복 구간에서는 조건문으로 제어 변수 `count`를 조사하여 반복 구간(`Print`)을 다시 호출할 수 있다.

01. 반복문이란?

LAB 6-1

정답

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  void Print(int count)
05  {
06      if(count > 0)
07      {
08          printf("난생처음 ");
09          Print(count - 1);
10      }
11  }
12
13  int main()
14  {
15      int n;
16      printf("반복 횟수를 입력하세요!\r\n");
17      scanf("%d", &n);
18
19      Print(n);
20  }
```

02

for문

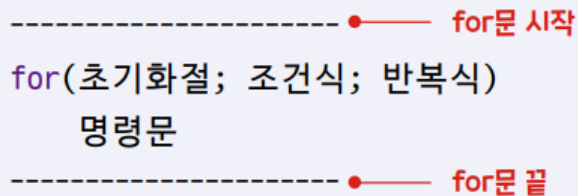
02. for문

I. for문의 개념

- for문

- 특정 횟수만큼 반복 실행하기 위하여 사용되는 반복문

- for문의 형식



The diagram illustrates the syntax of a for loop. It shows a sequence of three components in parentheses: an initialization statement, a condition, and a loop body. The first component is labeled 'for문 시작' (for loop start) with a red dot and arrow. The second component is labeled 'for문 끝' (for loop end) with a red dot and arrow. The third component is labeled '명령문' (statement). The components are separated by semicolons.

```
for(초기화절; 조건식; 반복식)  
    명령문
```

- 초기화절 : 초기화를 위한 변수 선언이나 초기화식
- 조건식 : 조건식의 계산 값이 0이 아니면(참) 명령문을 실행하고, 0이면(거짓) for문을 빠져나옴
- 반복식 : 명령문이 실행된 이후에 계산되는 식

02. for문

I. for문의 개념

- for문의 실행 흐름

1. 초기화절을 실행

2. 조건식 평가

- 참일 경우 : 명령문을 실행한 후 다시 for문의 반복식-조건식으로 돌아옴
- 거짓일 경우 : 즉시 for문을 빠져나옴

02. for문

I. for문의 개념

- for문의 실행 흐름

[코드 6-1] for문

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i;
06      for(i = 0; i < 2; i = i + 1)
07          print("%d\r\n", i);
08  }
```

0

1

02. for문

I. for문의 개념

- for문의 실행 흐름

[코드 6-2] 초기화절 변수 선언

```
01  #include <stdio.h>
02
03  int main()
04  {
05      for(int i = 0; i < 2; i++)
06          print("%d\r\n", i);
07      i = 1;
08  }
```

← 오류 발생

02. for문

I. for문의 개념

- for문에서 기억해야 할 점
 - for문의 진입이 최초인가 아닌가 여부
 - 최초 진입 시 : 초기화절, 조건식만 순서대로 실행 및 평가됨
 - 두 번째 진입 이후 : 반복식, 조건식만 순서대로 평가됨

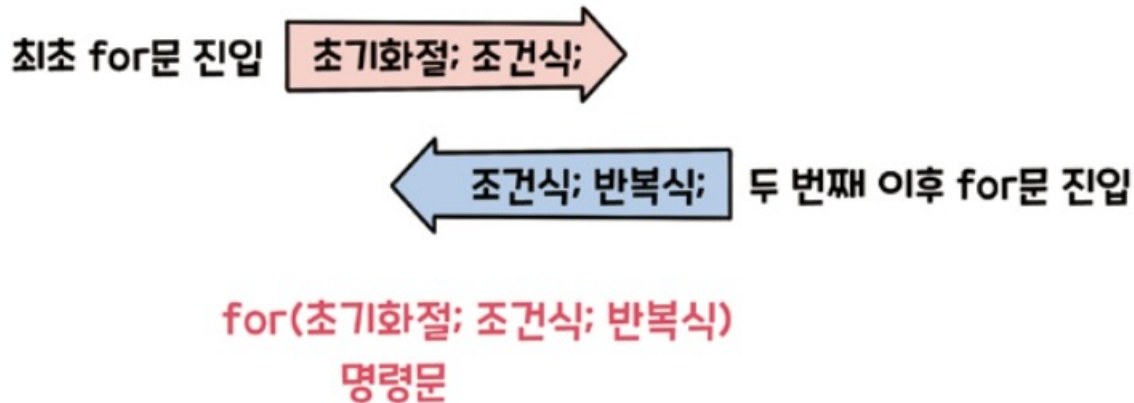


그림 6-2 for문의 실행 순서

02. for문

II. for문의 활용

- 특정 횟수만큼 반복하는 프로그램

[코드 6-3] for문을 이용한 합계

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int Sum = 0;
06      for(int i = 0; i < 100; i++)
07          Sum = Sum + (i + 1);
08
09      printf("총합: %d", Sum);
10  }
```

총합: 5050

02. for문

II. for문의 활용

- 특정 조건을 만족할 때까지 반복하는 프로그램

[코드 6-4] for문을 이용한 찾기

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 0;
06      int Sum = 0;
07      for(i = 0; Sum < 3000; i++)
08          Sum = Sum + (i + 1);
09
10      printf("1 +...+ %d >= %d", i, Sum);
11  }
```

1 +...+ 77 >= 3003

02. for문

II. for문의 활용

- 특정 조건을 만족할 때까지 반복하는 프로그램

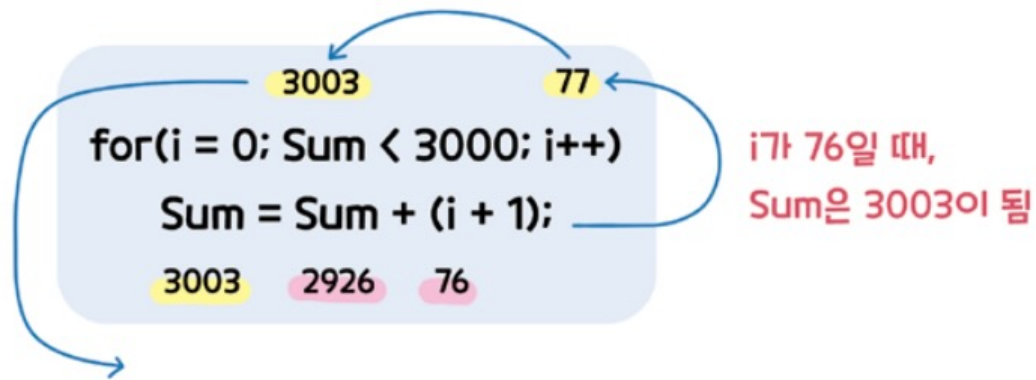


그림 6-3 [코드6-4] 반복문의 탈출 과정

02. for문

III. for문의 변형

- 초기화절과 반복식을 생략한 경우

[코드 6-5] 초기화절, 반복식 생략

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 0;                // 초기화절
06      int Sum = 0
07
08      for( ; i < 100; )
09      {
10          Sum = Sum + (i + 1);
11          i++;                  // 반복식
12      }
13
14      printf("총합: %d", Sum);
15  }
```

02. for문

III. for문의 변형

- 조건식을 생략한 경우

[코드 6-6] for문의 무한반복

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 0;           // 초기화절
06      int Sum = 0
07
08      for(;;)
09      {
10          Sum = Sum + (i + 1);
11          i++;             // 반복식
12
13          if(i > 99)
14              break;       // for문 탈출
15      }
16      printf("총합: %d", Sum);
17  }
```

02. for문

확인문제2

1. 다음 빈칸에 공통으로 들어갈 단어를 쓰시오.

C99 이후부터 for문의 초기화절에 변수를 할 수 있습니다. 초기화절에서 된 변수는 오직 for문 안에서만 이용할 수 있습니다.

2. 다음 for문에서 Code는 몇 번 실행되고, 그 이유는 무엇인지 설명하시오.

```
01  for(int i = 0; 1; i++)  
02  {  
03      Code;  
04  }
```

3. for문의 초기화절, 조건식, 반복식은 생략 가능하다. 만일 조건식이 생략된 경우 어떻게 평가되는가?

02. for문

LAB 6-2

팩토리얼 계산하기 1

$n!$ 의 계산 값이 1,000을 넘지 않는 최대의 n 과 그때의 $n!$ 을 구하고 출력하는 프로그램을 for문으로 작성해봅시다.

$$n! = 1 \times 2 \times 3 \times \dots \times n-1 \times n$$

6!: 720

- 1 반복 횟수를 모르기 때문에 for문 조건식은 생략하여 무한 반복한다.
- 2 for문 명령문 안에서 $n!$ 이 1,000을 넘을 때 for문을 탈출하고 그 때의 계산 값을 출력한다.

02. for문

LAB 6-2

정답

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i;
06      int factorial = 1;
07
08      for(i = 1; ; i++)
09      {
10          int next = factorial * i;
11          if(next >= 1000)
12              break;
13
14          factorial = next;
15      }
16
17      printf("%d!: %d", i - 1, factorial);
18  }
```

03

while문

03. while문

I. while문

- while문의 개념

- 반복문의 일종으로 오직 조건식만 가짐
- if문과 달리 조건식이 생략될 수 없음
- for문과 마찬가지로 반복되는 명령문은 단 하나만 올 수 있음

- while문의 형식

```
-----● while문 시작  
while(조건식)  
    명령문  
-----● while문 끝
```

03. while문

I. while문

[코드 6-7] while문

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 0;           // 초기화절
06      int Sum = 0;
07
08      while(i < 100)
09      {
10          Sum = Sum + (i + 1);
11          i++;             // 반복식
12      }
13
14      printf("총합: %d", Sum);
15  }
```

총합: 5050

03. while문

II. while문과 for문의 비교

- while문은 for문으로 완전히 대체 가능

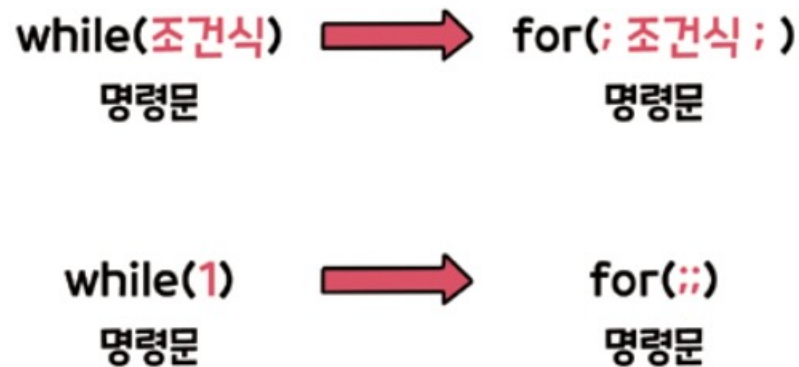


그림 6-4 for문으로 대체할 수 있는 while문

03. while문

II. while문과 for문의 비교

[코드 6-8] while문의 무한 반복

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 0;
06      int Sum = 0;
07
08      while(1)
09      {
10          Sum = Sum (i + 1);
11          i++;                // 반복식
12
13          if(i > 99)
14              break;          // while문 탈출
15      }
16
17      printf("총합: %d", Sum);
18  }
```

총합: 5050

03. while문

확인문제3

1. **while**문에는 초기화절이나 반복식이 없고 오직 조건식만 있다. 이때 특정 횟수를 반복하기 위해서는 어떻게 해야 하는지 서술하시오.
2. 다음 코드를 **while**문으로 변경하시오.

```
01  for(int i = 0; i < 100; i++)  
02  {  
03  }
```

03. while문

LAB 6-3

팩토리얼 계산하기 2

$n!$ 의 계산 값이 1,000을 넘는 최소의 n 과 그때의 $n!$ 을 구하고 출력하는 프로그램을 while문으로 작성해봅시다.

```
7!: 5040
```

while문의 조건식으로 factorial이 1,000보다 작은 경우 factorial을 반복해서 증가시키고 while문에서 빠져나올 때 결과 값을 출력한다.

03. while문

LAB 6-3

정답

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i = 1;
06      int factorial = 1;
07
08      while(factorial < 1000)
09      {
10          i++;
11          factorial *= i;
12      }
13
14      printf("%d!: %d", i, factorial);
15  }
```

04

do~while문

04. do~while문

I. do~while문의 형식

- do~while문의 개념
 - while문의 변형
 - 최소한 한 번은 명령문이 실행됨
- do~while문의 사용 형식

```
-----●----- do~while문 시작  
do  
    명령문  
while(조건식);  
-----●----- do~while문 끝
```

04. do~while문

II. do~while문의 활용

- do~while문이 활용되는 경우
 - 무조건 한 번은 사용자에게 입력을 받아야 함

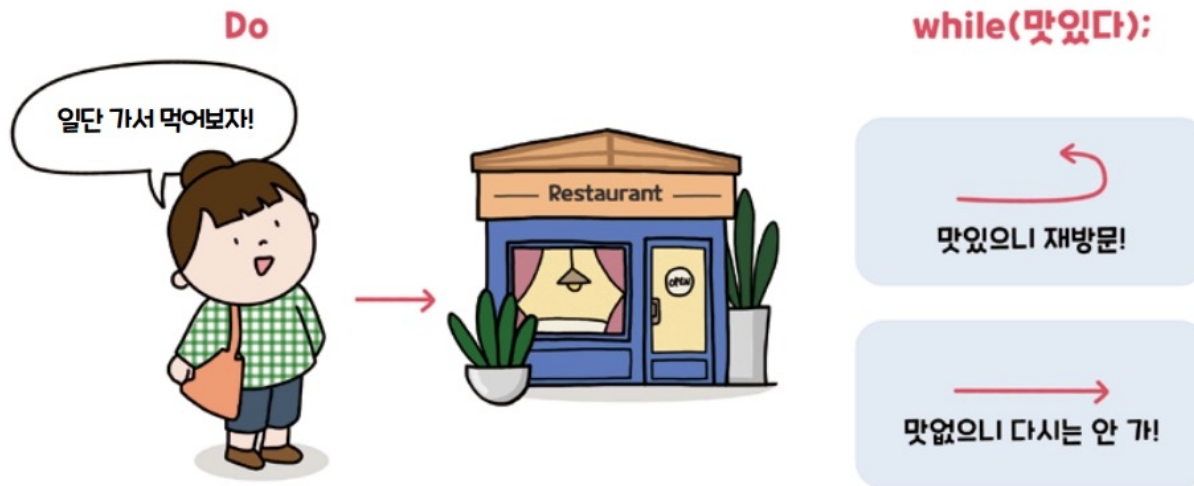


그림 6-5 do~while문이 활용되는 경우

04. do~while문

II. do~while문의 활용

[코드 6-9] while을 이용한 입력

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int Number;
07
08      printf("1번부터 4번까지 원하는 번호를 입력하세요! ");
09      scanf("%d", &Number);
10
11      while(Number > 4)
12      {
13          printf("1번부터 4번까지 원하는 번호를 입력하세요! ");
14          scanf("%d", &Number);
15      }
16
17      printf("선택한 번호는 %d입니다.", Number);
18  }
```

1번부터 4번까지 원하는 번호를 입력하세요! 7
1번부터 4번까지 원하는 번호를 입력하세요! 3
선택한 번호는 3입니다.

04. do~while문

II. do~while문의 활용

[코드 6-10] do~while을 이용한 입력

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int Number;
07
08      do
09      {
10          printf("1번부터 4번까지 원하는 번호를 입력하세요! ");
11          scanf("%d", &Number);
12      }
13      while(Number > 4);
14
15      printf("선택한 번호는 %d입니다.", Number);
16  }
```

04. do~while문

확인문제4

1. while문과 비교할 때 do~while문의 가장 큰 차이점은 무엇인지 서술하시오.
2. while문에서도 최소 1회 이상 명령문을 무조건 실행할 방법을 설명하시오.

04. do~while문

LAB 6-4

입력한 수를 모두 합하는 프로그램

입력한 수를 모두 합한 결과를 보여주는 프로그램을 작성해봅시다. 단, 0이 입력되면 결과를 보여줍니다.

```
더하고 싶은 수를 입력하세요 > 1  
더하고 싶은 수를 입력하세요 > 2  
더하고 싶은 수를 입력하세요 > 3  
더하고 싶은 수를 입력하세요 > 0  
총합은 6입니다.
```

- 1 do~while문을 활용하여 입력된 수가 0이 아닐 때까지 sum에 더한다
- 2 출력된 수가 0일 때 do~while문을 빠져나와서 결과를 출력한다.

04. do~while문

LAB 6-4

정답

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int n, sum = 0;
07
08      do
09      {
10          printf("더하고 싶은 수를 입력하세요 > ");
11          scanf("%d", &n);
12          sum += n;
13      }
14      while(n != 0);
15
16      printf("총합은 %d입니다.", sum);
17  }
```

05

반복문 고급

05. 반복문 고급

I. 다중 반복

- 다중 반복의 개념
 - 반복문 안에서 또 다른 반복문이 사용되는 것

[코드 6-11] for 다중 반복

```
01  #include <stdio.h>
02
03  int main()
04  {
05      for(int i = 2; i < 10; i++)    // 2단부터 9단까지
06      {
07          for(int j = 1; j < 10; j++)
08              printf("%d * %d = %d\r\n", i, j, i * j);
09          printf("\r\n");    // 단마다 줄 바꿈
10      }
11  }
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
...(생략)...
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

05. 반복문 고급

II. break

- 다중 반복문에서 break 사용 시 주의점
 - break는 오직 자신을 포함한 가장 가까운 반복문만 탈출함

[코드 6-12] 다중 반복과 break

```
01  #include <stdio.h>
02
03  int main()
04  {
05      for(int i = 2; i < 10; i++)
06      {
07          int j = 1;
08          while(1)
09          {
10              printf("%d * %d = %d\r\n", i, j, i * j);
11              j++;
12              if(j > 9)
13                  break;           //while문만 탈출함
14          }
15          printf("\r\n");
16      }
17  }
```

05. 반복문 고급

III. continue

- **continue의 개념**
 - 실행 흐름을 변경시킬 때 사용되는 명령어
 - break문 : switch문과 반복문에 사용됨
 - continue문 : 반복문에서만 사용됨
- **continue의 실행 흐름**

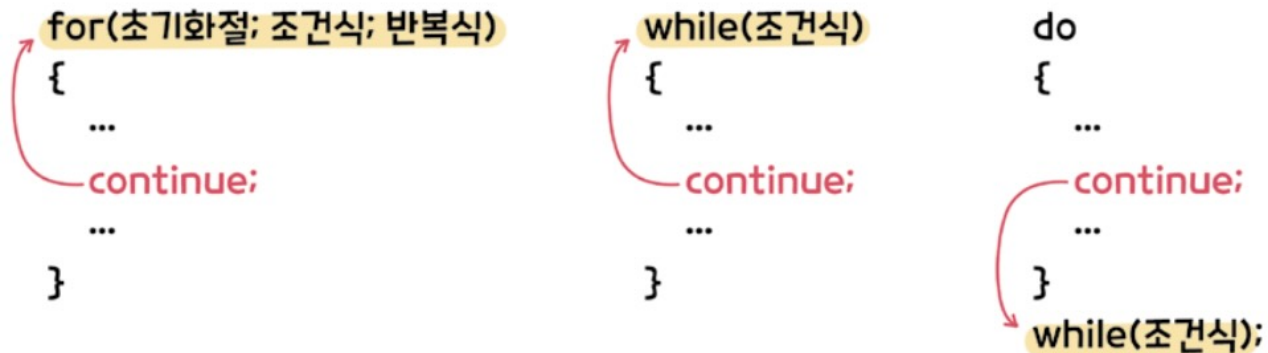


그림 6-6 continue의 실행 흐름

05. 반복문 고급

III. continue

[코드 6-13] continue

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int Number;
07
08      while(1)
09      {
10          printf("1번부터 4번까지 원하는 번호를 입력하세요! ");
11          scanf("%d", &Number);
12
13          if(Number > 4)
14          {
15              continue;
16          }
17          printf("선택한 번호는 %d입니다.", Number);
18          break;
19      }
20  }
```

1번부터 4번까지 원하는 번호를 입력하세요! 7
1번부터 4번까지 원하는 번호를 입력하세요! 3
선택한 번호는 3입니다.

05. 반복문 고급

확인문제5

1. 다음 코드의 7행에서 `break;` 뒤의 실행 흐름은 몇 행으로 변경되는지 설명하시오.

```
01  for(;;)
02  {
03      for(;;)
04      {
05          for(;;)
06          {
07              break;
08          }
09      ① 지점A
10      }
11      ② 지점B
12  }
13      ③ 지점C
```

2. 다음 진술의 참/거짓을 판별하고, 그 이유를 서술하시오.

`continue`를 만나면 반복문의 시작 지점으로 실행 흐름이 돌아간다.

05. 반복문 고급

LAB 6-5

생년월일 입력받기

생년월일을 차례대로 입력받아서 보여주는 프로그램을 작성해봅시다. 이때 제대로 된 숫자(연 2021 이하, 월 12 이하, 일 31 이하)가 입력되지 않을 경우 반복해서 입력을 요구하며, 단 하나라도 0 이하 수가 입력되면 프로그램을 바로 종료합니다.

태어난 연도를 입력하세요 > 2100

태어난 연도를 입력하세요 > 2010

태어난 월을 입력하세요 > 3

태어난 일을 입력하세요 > 29

Birthday: 2010-3-29

- 1 scanf를 통해서 태어난 연도, 월, 일을 입력받고 유효하지 않은 수가 입력되었을 경우 continue를 통해서 반복문 처음으로 돌아가서 재입력을 요구한다.
- 2 일 반복문을 월 반복문이 포함하며, 월 반복문은 연도 반복문에 포함된다.
- 3 반복문 중에서 단 하나라도 0 이하의 수가 입력될 경우 return 0;을 통해서 main 함수를 즉시 반환하여 프로그램을 종료한다.

05. 반복문 고급

LAB 6-5

정답

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int year, month, day;
07
08      while(1)
09      {
10          printf("태어난 연도를 입력하세요 > ");
11          scanf("%d", &year);
12
13          if(year > 2021)
14              continue;
15          else if(year <= 0)
16              return 0;
17
```

05. 반복문 고급

LAB 6-5

정답

```
18     while(1)
19     {
20         printf("태어난 월을 입력하세요 > ");
21         scanf("%d", &month);
22
23         if(month > 12)
24             continue;
25         else if(month <= 0)
26             return 0;
27
28         while(1)
29         {
30             printf("태어난 일을 입력하세요 > ");
31             scanf("%d", &day);
32
```


05. 반복문 고급

LAB 6-5

정답

```
33         if(day > 31)
34             continue;
35         else if(day <= 0)
36             return 0;
37
38         printf("Birthday: %d-%d-%d", year, month, day);
39         return 0;
40     }
41 }
42 }
43 }
```

[실전예제]

100 이하의 소수 출력하기

[실전예제] 100 이하의 소수 출력하기

[문제]

100 이하의 소수를 모두 출력하는 프로그램을 작성해봅시다.



실행 결과

100 이하의 소수: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

[실전예제] 100 이하의 소수 출력하기

[해결]

1. 소수는 1과 자기 자신으로만 나누어지는 수이다.
따라서 가장 작은 소수는 2이다.
2. 2를 제외한 짝수는 모두 2로 나누어지므로 소수가 아니다.
따라서 3 이상 100 이하의 홀수를 후보로 3부터 후보보다 작은 수까지 순서대로 나누어 떨어지는지 검사하여 소수 여부를 확인한다.
3. 만일 나누어 떨어진다면 소수가 아니므로 break를 통해서
안쪽 for문을 탈출한 후 후보를 2 증가시킨다.
4. 3부터 후보보다 작은 수까지 순서대로 나누어 떨어진 적이 없다면
자연스럽게 안쪽 for문을 빠져나왔으므로 소수이다.

[실전예제] 100 이하의 소수 출력하기

[해결]

```
01  #include <stdio.h>
02
03  int main()
04  {
05      printf("100 이하의 소수: 2, ");
06
07      for(int i = 3; i <= 100; i += 2)
08      {
09          int flag = 1;
10          for(int j = 3; j < i; j += 2)
11          {
12              if(i % j == 0)
13              {
14                  flag = 0;
15                  break;
16              }
17          }
18
19          if(flag)
20              printf("%d, ", i);
21      }
22  }
```

Thank you!