**Lab 01 – Basic of C Programming**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 프로그래밍 언어 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.03.04.목 |

1. **Pseudo Code**

**Procedure find\_max\_min\_avg(void)**

1: // input argument : none

2: int nums[5];

3: int max, min;

4: double avg; // 평균은 실수값으로 선언

5: input nums[0], nums[1], nums[2], nums[3], nums[4];

6: max = nums[0]; // 초기값을 주어진 데이터의 첫번째 값으로 정의

7: min = nums[0];

8: avg = nums[0];

9: int i = 0;

10: while (i < 5)

11: {

12: if (max < nums[i]) max = nums[i];

13: if (min > nums[i]) min = nums[i];

14: avg = avg + (double)nums[i];

15: i = i + 1;

15: }

16: avg = avg / 5;

17: printout max, min, avg;

**End of Procedure find\_max\_min\_avg(void)**

1. 유사코드를 한 줄씩 실행할 경우, 지역변수들이 어떤 값을 가지게 되는지 나타내어라(input 3, 2, 5, 1, 4)

주로 관찰해야할 지역 변수로는 max, min, avg가 있다.

각 데이터를 입력받을 때의 변화를 표로 작성해 보면

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 3 | 2 | 5 | 1 | 4 |
| max | 3 | 3 | 5 | 5 | 5 |
| min | 3 | 2 | 2 | 1 | 1 |
| avg | 3 | 5 | 10 | 11 | 15 |

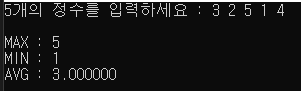
avg는 마지막에 평균을 구하기 위해 5로 나눈 값인 3이 된다

1. 작성한 유사코드로 정확하게 계산되는지 확인하라

유사코드를 통해 결과값을 도출했을 경우 최대값은 5, 최소값은 1, 평균값은 3이다.

이는 2번 문항의 값과 동일함을 알 수 있다. 따라서, 유사코드를 통한 결과는 정확하다는 것을 알 수 있다.

실제로, 이 유사코드로 작성한 C 프로그램 실행창을 확인하면 아래의 값이 나온다.



1. 작성한 유사코드를 기반으로 C 프로그램을 작성하라

/\*

\* 파일명 : find\_max\_min\_avg.c

\* 목적 및 기본 기능 : 입력받은 정수 데이터의 최대값, 최소값, 평균값을 산출

\* 작성자 : 김주환(21812158)

\* 작성일 : 2021년 3월 4일

\*/

#include <stdio.h>

int main(void) {

int nums[5] = { 0 };

printf("5개의 정수를 입력하세요 : ");

scanf\_s("%d %d %d %d %d", &nums[0], &nums[1], &nums[2], &nums[3], &nums[4]);

int max = nums[0];

int min = nums[0];

double avg = nums[0];

int i = 0;

while (++i < 5) {

if (max < nums[i]) max = nums[i];

if (min > nums[i]) min = nums[i];

avg += (double)nums[i];

}

avg /= 5;

printf("\nMAX : %d\nMIN : %d\nAVG : %lf\n", max, min, avg);

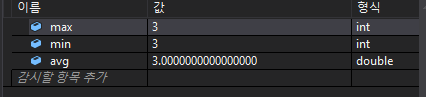
return 0;

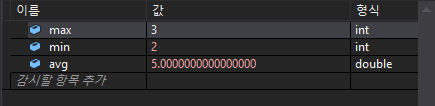
}

1. Visual studio의 debugger 기능을 사용하여 실행 중간 상태에서 지역변수들의 값이 정확하게 변경되는지 확인하라

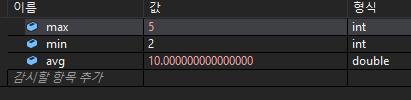
F5, F10, F11 모두 각 변수가 변하는 지점에서 아래의 변화를 가진다.

nums[0];

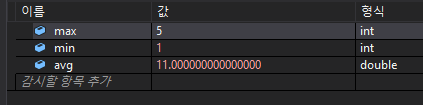
  
nums[1];



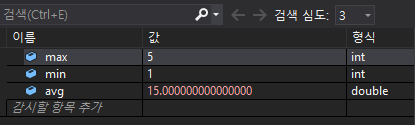
nums[2];



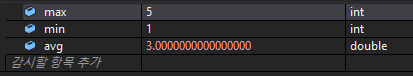
nums[3];



nums[4];



Avg /= 5;



Oral Test

|  |
| --- |
| **(1) 소프트웨어 개발단계에서의 설계 및 구현 상세 절차에 대하여 설명하고, 유사코드가 무엇이며, 언제 작성하는 것인가에 대하여 설명하라.**  소프트웨어 개발단계는 크게 2가지 부분으로 나뉜다. 설계와 구현이다. 설게 단계에서는 주어진 문제를 잘 분석하고 문제에 대한 정확한 이해 그리고 요구사항을 정확히 파악해서 관련된 소프트웨어를 구현해야한다. 분석이 끝나면 주어진 문제를 해결하는 데에 가장 적합한 알고리즘과 자료구조를 설계한다. 이 과정에서 문제를 어떻게 해결할 것인지가 정해진다. 이후 시스템을 설계한다. 이때 전체 Software system을 설계하고 이를 나누어서 Subsystem 별로 부분적으로 나누어서 설계하고 이를 다시한번 상세하게 나누게 되면 Block과 Module로 나누게 된다. 가장 작게 나눈 기본적인 단위가 Class 또는 Function이다. 이상의 설계를 기반으로 구현단계로 넘어가게 됩니다.  이제 소스코드를 작성합니다. 소스코드를 작성하는 중간중간 파일을 저장하게 되는데 사용하는 언어에 따라서 다른 확장자를 사용하게 됩니다. 이렇게 작성된 소스코드는 부분적으로 컴파일을 할 수 있습니다. 이때 문법적인 오류가 있으면 컴파일러를 통해 나타나게 되고 수정과정을 거치게 됩니다. 문법적인 오류를 모두 해결하고 나면 Object codes가 생성 된다. 실제로 프로그램을 실행시키면 Object codes가 아니라 Executable코드 형태로 구성됩니다. 이는 링크를 통한 결합으로 만들어진다. 실제로 프로그램을 실행시키려면 실행코드를 Load를 통해서 메모리에 장착을 하고 cpu가 실행하게 된다. 실행된 결과가 출력되면 결과를 보고 원한 값이 맞는지 확인을 한다. 이때 오류가 있다면 부분적으로 소스코드를 수정 후 다시 시행하게 된다. 또한 가장 큰 문제가 발생하는 부분들이 알고리즘이나 자료구조의 설계를 처음부터 다시해야하는 문제가 발생할 수 있다. 예를들어 요구되는 시간이 굉장히 짧거나, 제공되는 하드웨어의 메모리 용량이 부족한 경우가 있다. 이런 경우를 피하기 위해 설계 부분을 아주 면밀하게 살펴보고 더 많은 시간을 투자해야한다.  유사코드{Pseudo code}: 프로그래밍 언어의 모든 문법을 정확하게 지키지 않고, 전체 알고리즘의 기본 구조와 기능을 간략화된 표현으로 설명하는 코드다.  유사코드는 개발 초기 소스 코드의 뼈대를 만들 때 작성한다. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **(2) C program이 컴파일 단계에서 발생할 수 있는 compile error message 에는 어떤 종류들이 있는지에 대하여 20가지 이상 예를 들어 표로 만들고 오류 수정 방법에 대하여 설명하라.**   |  |  |  | | --- | --- | --- | | C 프로그램 문법적 오류 | 오류 메시지 | 올바른 입력 | | #include <stdjo.h> | C1083 포함파일을 열 수 없습니다.  ‘stdjo.h’: No such file or directory | #include <stdio.h> | | #include stdio.h | C2006: ‘#include’: expected  “FILENAME” of <FILENAME> | #include <stdio.h> | | int x  double y; | C2144: 구문오류: double은 ‘;’다음에 와야 합니다. | int x;  double y; | | retun 0; | C2065 ‘retun’; 선언되지 않은 식별자입니다 | return 0; | | main()  {. . . .} | C4430: 형식지정자가 없습니다.  int로 가정합니다. | int main()  {return 0;} | | void main()  int x; | E0130 ‘{‘가 필요합니다.  C2144: int는’;’다음에 와야합니다. | void main()  { int x; } | | printf(“”); | C3861: ‘printf: 식별자를 찾을 수 없습니다. | #include <stdio.h>  printf(“”); | | it x; | E0020 식별자 “it”가 정의되어있지 않습니다. | int x; | | int x;  printf(“%d”); | C6001 초기화되지 않은 메모리 ‘x’를 사용하고 있습니다. | int x = 0;  printf(“%d”); | | int x = 1;  scanf(“%d”, x); | C6066 ‘scanf’에 대한 호출에는 포인터가 필요합니다. | int x = 1;  scanf(“%d”, &x); | | print(“”); | C4013 ‘print’가 정의되지 않았습니다. | printf(“”); | | scanf(); | E0165 함수 호출에 인수가 너무 적습니다. | scanf(“%d”); | | int x=1;  int arr[x]; | E0028 식에 상수 값이 있어야 합니다. | int arr[1]; | | for (int i = 0, i < 5, i++); | E0065 ‘;’가 필요합니다. | for (int i = 0; i < 5; i++); | | for int i = 0; i < 5; i++; | E0125 ‘(‘가 필요합니다. | for (int i = 0; i < 5; i++); | | int x = 1;  x /= 0; | C4723 0의 나누기 연산이 발생할 수 있습니다. | int x = 1;  x /= 1; | | int printf; | C4101 ‘printf’: 참조되지 않은 지역변수 입니다. | int x; | | int 4; | E0040 식별자가 필요합니다. | int a; | | int arr[1] = { 0, 1 }; | E0146 이니셜라이저 값이 너무 많습니다. | int arr[1] = { 0 }; | | int x = 20000000000; | C4305 ‘초기화중’: ‘\_\_int64’에서 ‘int’로 잘립니다. | int x = 2000000000; | |

|  |
| --- |
| **(3) Visual studio의 debugger 기능인 F5, F10, F11에 대하여 설명하라.**  디버깅은 논리적인 오류를 해결하기 위해 사용하는 기능이며 용도에 따라 다음으로 나뉜다.  F5 (+F9): 기본적인 디버깅 기능이며 F9와 함께 사용하면 원하는 라인에서 중단하여 그 사용성을 높여줄 수 있다. 이를 실습에 적용하게 되면 디버깅을 할 때 필요한 부분은 지역변수가 변화되는 부분이기 때문에 각 반복문의 시작이나 끝에 중단점을 찍고 마지막에 평균을 구할 때도 중단점을 찍어서 사용할 수 있다.(주의: 중단점에 해당하는 라인은 아직 실행되지 않았음)  F10: 라인 바이 라인으로 디버깅을 진행하며, 함수 내로는 진입하지 않습니다. 이를 실습에 적용하게 되면 F5와는 다르게 라인 바이 라인 디버깅이기 때문에 지역변수의 변화 보다는 코드작성 중의 실수를 찾아내는 방향으로 사용할 수 있겠다.  F11: 라인 바이 라인으로 디버깅을 진행하며, F10과 다르게 함수 내로 진입합니다. 이를 실습에 적용하기에는 해당 실습 문제에 함수를 사용하지 않았기 때문에 F10과 같은 역할을 수행할 수 있겠다. |

|  |
| --- |
| **(4) Visual studio의 debugger 기능을 사용하여 프로그램의 실행 중간 단계에서 지역변수들의 값이 어떻게 변하는지 확인할 수 있는 방법에 대하여 설명하라.**  크게 자동, 로컬, 조사식의 3가지 방법을 통해 각 지역변수들의 값 변화를 확인 할 수 있습니다.  첫번째로는, 자동 방식입니다. 이는 디버깅 과정에서 자동으로 해당 라인(또는 중단점) 주위의 변수를 표시합니다.  두번째로, 로컬 방식입니다. 이는 디버깅 과정에서 현재 함수 또는 로컬 범위에 정의된 변수를 표시합니다.  세번째로, 조사식 방식입니다. 이는 디버깅 과정에서 원하는 변수를 지정하고, 지정된 변수를 표시합니다.  위의 세가지 방식을 통해 실습 1의 프로그램의 각 변수들이 어떻게 변화되며 최종적으로 어떤 값이 되는지 직관적으로 확인할 수 있습니다.  (추가로 위의 기능이 보이지 않는다면 디버그 -> 창 -> 조사식/ 자동/ 지역 순으로 들어가서 확인 할 수 있습니다.) |