|  |
| --- |
| **[506489] 시스템프로그래밍** |
| **실습 #03 문제 및 보고서** |

|  |  |
| --- | --- |
| **이름** | 곽영주 |
| **학번** | 20175105 |
| **소속**  **학과/대학** | 빅데이터 |
| **분반** | 01 (담당교수: 김태운) |

## <주의사항>

* 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
* 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
  + 소스코드/스크립트 등을 작성 한 경우, 해당 파일의 이름도 적어주세요.
* SmartLEAD 제출 데드라인:
  + 월요일 분반: 다음 실습 시간 전날까지(일요일 까지)
  + 수요일 분반: 다음 실습 시간 전날까지 (화요일 까지)
  + 데드라인을 지나서 제출하면 24시간 단위로 20%씩 감점(5일 경과 시 0점)
  + 주말/휴일/학교행사 등 모든 날짜 카운트 함
  + 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
  + 예외 없음
* SmartLEAD 에 아래의 파일을 제출해 주세요
  + 보고서(PDF 파일로 변환 후 제출 권장하나, WORD 형식으로 제출도 가능)
  + 보고서 파일명에 이름과 학번을 입력해 주세요.
  + 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일 제출 (zip 파일로 압축하여 하나의 파일로 제출)

## <개요>

이번 과제는 3주차 강의 내용을 복습하는 내용입니다. 3주차 강의 내용은 리눅스 운영체제에서 사용하는 프로그램 개발 환경에 관련된 내용입니다 (gcc 컴파일러, Makefile 등).

## <실습 과제>

|  |
| --- |
| **[Q 0] 요약 [20점]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

답변: 이번 과제를 통해 Makefile 사용법과 매크로 사용법을 익혔고, malloc, calloc, realloc을 사용한 동적 메모리 관리를 배웠습니다. 또한, 명령행 인자에 대해 개념이 제대로 잡히지 않았는데 이번 과제를 통해 개념을 바로 잡을 수 있었습니다.

|  |  |  |  |
| --- | --- | --- | --- |
| **[Q 1] Makefile [20점]**  int형 정수를 입력으로 받아, 덧셈, 뺄셈, 곱셈 연산을 수행하는 계산기 프로그램을 개발하세요. 아래와 같이 총 4개의 소스코드를 작성해야 합니다.   * ‘calc.c’ : 계산기 구동 프로그램(main 함수 구현) * ‘add.c’ : 덧셈 연산을 수행하는 함수 구현 * ‘sub.c’ : 뺄셈 연산을 수행하는 함수 구현 * ‘mul.c’ : 곱셈 연산을 수행하는 함수 구현   총 4개의 소스코드를 컴파일 해서 calc.exe 라는 실행파일을 생성하기 위해, Makefile 을 만드세요. Makefile 에는 최소한 2개의 target이 정의되어 있어야 합니다. 하나는 ‘all’ 이고, 나머지 하나는 ‘clean’ 입니다.  계산기 프로그램을 실행하면 아래와 같이 터미널에 출력합니다:   |  | | --- | | <Calculator Menu>  1. Add  2. Sub  3. Mul  Enter : |   1, 2, 3 중에서 하나의 숫자를 입력 후 엔터를 누르면 다음가 같이 출력합니다.   |  | | --- | | Enter two numbers : |   공백으로 구분된 두개의 int형 숫자를 입력 후 엔터를 누르면 연산 결과가 아래와 같이 출력됩니다. (예: 덧셈을 수행하는 경우)   |  | | --- | | Result : 12 + 34 = 46 |   프로그램 실행 화면을 캡처하고, 본 문서에 첨부하세요 (3가지 연산에 대해 각각 터미널 화면을 캡처하세요). 소스코드 및 Makefile을 제출하세요. |

답변 (터미널 화면 캡처):

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| **[Q 2] 명령행 인자 [20점]**  ‘getopt’ 함수를 사용해서 다음 명령 및 인자/옵션을 처리하는 프로그램을 작성하세요.   * 명령 이름(=프로그램 이름=실행파일 이름) : hallym * 옵션 설명:   + 인자가 주어지지 않은 경우 : “ERROR: please provide at least one option.” 메시지를 터미널에 출력   + -a : Welcome to System Programming (2019-2) 메시지를 터미널에 출력   + -u 인자 : Nice to meet you, 인자 와 같이 터미널에 출력   + -h : 사용 가능한 옵션 목록을 터미널에 출력 (출력 양식/형식은 자유)   소스코드를 제출하세요. 아래와 같이 각각을 실행하고, 터미널 출력 결과를 캡처해서 본 문서에 첨부하세요.  1) 인자가 주어지지 않은 상태로 실행  2) –a 인자와 –u 인자가 동시에 주어진 상태로 실행  3) –h 인자만 주어진 상태로 실행 |

답변 1):

텍스트이(가) 표시된 사진

자동 생성된 설명

답변 2):

텍스트이(가) 표시된 사진

자동 생성된 설명

답변 3):

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| **[Q 3] 동적 메모리 관리 [20점]**  동적으로 배열의 크기를 늘리고 줄이는 프로그램을 개발하는 문제입니다.  ‘main’ 함수에는 아래의 변수가 선언되어 있습니다.  int size = 10; // int형 변수를 저장하는 배열 arr 의 크기 (기본값)  int n = 0; // arr 배열에 저장된 int 형 데이터 개수  int\* arr; // 배열  int sum = 0; // 배열에 저장된 숫자들의 합  ‘size’ 개수의 정수를 저장할 수 있도록 메모리를 동적으로 할당하고, 동적으로 할당한 배열을 가리키는 포인터를 arr 변수에 저장하세요. 현재, size 변수에는 10이라는 값이 저장되어 있습니다. 10은 size변수의 초기값이며, 동시에 최소값 입니다. size변수에 저장되는 값은 동적으로 변하지만, 10보다 작아질 수는 없습니다.  이제, n, size, sum 값을 터미널에 출력하세요.  반복 문을 사용해서 1부터 100까지 모든 정수를 arr 배열에 순차적으로 저장하세요. 배열에 숫자가 한 개 추가될 때 마다 n 값은 1씩 증가합니다. 배열에 더 이상 공간이 없는 경우, 배열의 크기를 2배로 만드세요. 배열의 크기가 증가하면 size 값도 따라서 증가해야 합니다. 배열이 2배로 증가할 때 마다, 다음과 같이 출력하세요:  printf(“Size up: %d => %d\n”, <변경 전 size 값>, <변경 후 size 값>).  100까지 정수를 저장한 뒤, arr 배열에 저장된 모든 숫자의 합을 sum에 저장하세요. 이제, n, size, sum 값을 터미널에 출력하세요.  반복 문을 사용해서 배열에서 숫자 100부터 11까지 순차적으로 삭제하세요 (= 배열에서 해당 숫자를 0로 만드세요). 정수 한 개를 삭제할 때 마다 n도 1씩 감소합니다. 만약 n <= size/4 이면, 배열의 크기를 반으로 줄이세요. 배열의 크기가 감소할 때 마다, 터미널 화면에 다음과 같이 출력하세요: printf(“Size down: %d => %d\n”, <변경 전 size 값>, <변경 후 size 값>). 배열의 크기를 줄일 때는 size 값도 업데이트 되어야 합니다. 참고로, size는 10보다 작아질 수 없습니다. 배열 arr에 저장된 모든 숫자의 합을 더하고 sum에 저장하세요.  이제, n, size, sum 값을 터미널에 출력하세요.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 작성한 소스코드를 제출하세요.  \*\* 주의: 동적으로 할당한 메모리 공간은 반드시 free 함수를 호출하여 해제해야 합니다. |

답변 (터미널 화면 캡처):

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| **[Q 4] DEBUG 매크로 [20점]**  위 [Q 3]에서 작성한 프로그램을 아래와 같이 수정하세요.   * 만약 DEBUG 라는 매크로가 정의되어 있는 경우, ‘변경 전 size 값’, ‘변경 후 size 값’을 출력하는 printf 구문을 사용하고, * 만약 DEBUG 매크로가 정의되어 있지 않으면, ‘변경 전 size 값’, ‘변경 후 size 값’을 출력하는 printf 구문을 사용하지 않음   DEBUG 라는 매크로는 소스코드 내에서 정의하지 않고, gcc 명령어 사용 시 해당 매크로를 정의하거나 또는 정의하지 않도록 결정합니다.  Makefile을 만들고, 아래와 같이 4개의 target을 정의하세요   * ‘all’ : 아래의 ‘release’ 타겟과 동일하게 동작함 * ‘debug’ : DEBUG 매크로를 정의하여 컴파일 * ‘release’ : DEBUG 매크로를 정의하지 않고 컴파일 * ‘clean’ : 소스코드/Makefile을 제외한, 불필요한 파일 삭제(예: \*.out 파일 등)   Makefile 본문을 복사하여 아래에 붙여 넣으세요. |

답변 (Makefile 본문):

텍스트이(가) 표시된 사진

자동 생성된 설명

**끝! 수고하셨습니다 ☺**