제목: C언어로 작동하는 고속 블록 코딩 프로그램 / 개선 제목: 블록 코딩 프로그램

작성자: 장용호, 팀: 11팀, 발표 시기: 2024년 12월 10일

작성목적: 해당 보고서는 이전 보고서가 개선된 프로그램을 담지 못했다는 점에 의거 새로운 보고서를 작성하기로 하였습니다.

초론: 해당 응용프로그램은 블록을 드래그하여 코딩을 하는것을 목표로 하였으며, 초보자도 쉽게 코딩을 할수가 있도록 제작하기로 계획을 하였다. 하지만 해당 프로젝트의 진전은 상당히 느렸으며 최근 시기에 중요한 기능들이 작성이 되었다. 해당 보고서는 이러한 기능들을 담고 작성하기 위해서 작성하였다.

간략 전개:

- 프로젝트의 구조체들 고안, 블록 구조체는 초기에는 다음 블록의 숫자 포인터와 함수의 포인터를 가지는 구조체로 고안이 되었으며 이 시기에 블록 구조체를 담는 컨테이너 구조체에 대해서 완성함

- SDL 라이브러리를 이용해서 윈도우를 제작 루프문을 구성하여 해당 윈도우의 작동 구조를 확립하였고 이때 다중 윈도우 기술에 대해서 시범 테스트를 함(이때 gpt 도움 받음)

- 윈도우에 대한 상대 x, y 정보를 저장하는 목적으로 SDL_Rect MainWindow 구조체를 선언 하는 등 전역 구조체들을 상당수 선언하였으며 SDL 드로우 부분에서 개선을 하였고 블록 드래그 등과 텍스트 그리기를 구현함

- sub.c 파일을 만들어서 지금의 Sum 블럭의 함수를 구현하였으나 테스트를 못하는 상태로 방치가 됨(근데 이후에 잘 작동함) (그리고 이후에는 gpt를 잊었다가 DB에서 부름)

- 처음에는 스페이스로 다음에는 우클릭으로 블록을 생성하는 식이었으나 이후 개편을 진행해서 UIRect 구조체를 제작하고 그것을 담을 구조체를 제작, 이후 관련 함수들을 제작함, UIRect은 초기의 블록과 같은 형태로 함수 포인터를 가지고 있어서 클릭시 포인터를 실행하거나 그냥 NULL로 놔둘수가 있음, 또한 블럭의 이름을 정하는 기능을 추가하였으며 Sum 블럭의 경우에는 플래그를 바꾸는 복합합 스위치 구문을 작성하였음

- 블럭 구조체를 개선하여 현재의 모습으로 변경, 타입과 태그로 블록의 메소드를 추측을 하며 해당 메소드인 함수를 불러와서 실행하는 방식으로 개편하였음, 이때에 print 블럭과 상수 블럭들을 구현하였으며 이전 보고서의 진도임, 이후 Sum 블럭을 함수와 연결하여서 작동시키도록 하였음

- c버튼 클릭으로 해당 블록을 카피하는 기능을 추가하고 이를 변수 블록에 할시 해당 블록은 해당 블록의 포인터 블록으로 작성, 이는 변수 기능을 추가하였음을 뜻함, 이후 함수 부분을 작성하였으나 미완성인 상태에서 저장 안한채 끝남, 다른 사람 수업 시간에 짬짬히 같은 시간을 들여 다시 제작하였음

위의 제작 과정은 다음과 같고 여기서 설명하지 못한 부분은 다음장에서 설명함
보충할점: (간략이니 코드 대응은 아님)

현재의 블록 구조체는 다음과 같은 형식으로 구성이 됨:

- 위치: 컨테이너에 저장 or id로 호출
- 생성: new_Block 함수로 생성하고 리턴은 id, get_Blcok 함수로 불러와서 수정
- 이름: new_str 함수로 생성하고 그 id를 저장
- 함수: type와 tag로 런타임에서 추측해서 실행
- 데이터: 값을 저장함
- next: 다음 블록을 지정함
- arg: 인수 블록을 지정함

현재의 컨테이너 구조체는 다음과 같은 형식으로 구성이 됨:

- 위치: 정적 (main) 과 동적 (main 이후)
- 생성: 메인은 전역 선언, 이후 노드는 new_Container 함수로 생성
- 저장: 블록 구조체와 스트링을 저장함
- 확장: next로 포인터에 동적으로 할당함
- 저장: db로 저장하고 불러옴

쓰이는 키는 다음과 같음:

- Sum: + - * / % == != < <= > >= (키 2개는 2번 입력)
- Block: 이름 입력할때 아스키 입력 가능
- Num: 숫자 상수 입력시는 숫자 입력만 됨
- Str: 텍스트 상수 입력은 블록 입력과 동일함
- c: c를 클릭하면 마지막에 접촉한 블록을 카피하여 생성함
- pointer: Var와 Event의 경우에는 그 블록을 참조하는 포인터를 생성함

실수한점:

- 이진 탐색 방식을 이용한 스텍을 구상을 하였음, 이 스텍은 중간에 값을 제거하는 것이 가능하였으며 탐색 방식상 가장 좌측의 빈 값을 우선적으로 채웠음, 허나 해당 스텍의 탐색 방식은 쓸데없이 거창했고 정작 사용 목적인 변수에 대해서는 블록의 데이터를 참조하는 형식으로 제작이 결정이 되었기에 쓸데없는 흔적 기관으로 남겼고 탐색 방식은 for 방식으로 개편이 되어서 지금은 SubWindow의 텍스트 출력에 쓰이는 텍스트를 저장하는 저장기관으로 쓰이고 있음

- SQLite를 이용한 바이너리 저장 방식을 사용해서 컨테이너 구조체를 저장하는 방식으로 프로그램을 끄면 저장하고 키면 불러오는 방식을 구현을 하였으나, 정작 해당 방식에서는 한번 저장하고 한번 불러오면 끝이고 그 다음에는 알수가 없는 오류로 저장된 바이너리가 비어있기에 불러오지를 못함 (빈 공간을 불러옴), 시기상 이전 보고서를 작성하기 직전인데 실이라 따로 적음

- 프론트엔드 부분에서에 너무 많은 시간을 소모하였음, 차라리 gpt에게 맡겼다면 모를까 일일히 작성을 하였기에 많은 시간을 소모한것 같음 (다만 SDL 실력은 많이 올랐음) 덕분에 정해진 기간 이내에 프로그램이 구색을 갖추지를 못하였음

기타 등등:

클릭: 클릭은 3가지의 경우로 나뉨

- 바탕화면 클릭: 바탕화면을 따라서 드래그함
- UIRect 클릭: 해당 블록에 지정된 함수 실행
- block 클릭: 해당 블록을 드래그함
- block 우클릭: 이름 변경 or Sum의 경우에는 연산자 변경
- main 클릭: 진입점을 해당 블록으로 변경 (구현했지만 저장 안됨)

컴파일: 블록을 해석하고 실행하는 것인데, 컴파일이라기에는 부실함

- 시작점으로 main 블록 선택
- 블록 해석 -> naxt 블록으로 옮김
- 함수 포인터 블록의 경우에는 해당 함수 호출
- 호출시에는 포인터의 next를 스텍에 푸시, NULL 발견시 스텍 pop

코드 해석: 코드 부분에 대해서 읽고 해석하는 부분임, 텍스트로 작성하고 그걸 보는것 보다는 코드 읽는 것이 훨씬 빠르겠지만 보고서에 간략하게라도 작성하는 것이 좋아 보임

시작점:

전역 구조체와 변수들을 선언함

메인 윈도우와 서브 윈도우를 생성하고 서브 윈도우를 숨김

메인 윈도우의 루프문을 실행하고 종료 플래그시 루프문을 break함

루프문에는 _Event 함수라는 이벤트 문이 있으며 스위치 구문으로 실행

스위치 구문에서는 여러가지 플래그에 대응하도록 함

- 종료 플래그: 종료함
- 마우스 클릭 업 다운 플래그: 클릭 전용
- 텍스트 인풋 이벤트 플래그: 입력 전용 (SumClick과 BlockNameClick에 사용)
- 키 플래그: 카피 구문이 있음

이런식으로 루프하는게 끝이기는 한데 UIRect 클릭 -> 업로드 블록의 함수에는 sub.c의 subWindow를 호출하는 함수를 호출하고 거기를 이후 서술하겠음

subWindow를 보이고 맨 앞으로 놓음

컴파일 함수를 실행

mainWindow의 루프 -> 서브윈도우 이벤트 여부 -> 서브윈도우 루프 함수 호출 -> 출력 버퍼에 저장 된 텍스트를 보임

그리고 부가 설명으로 컴파일 함수는 간략하게 서술해서

main 블록의 id를 지역 변수 getch_id에 저장

getch_id가 0이라면 (다음 블록이 없을때를 가정)
-함수 호출 스텍 pop
-값이 0이라면 컴파일 끝

else 이라면

블록을 블록 해석기에 박음

블록 해석기는 블록의 함수를 추측하여 실행함

그리고 next 블록을 다음 id로 지정

블록의 함수는... 해당 블록의 메소드임

Sum은 플래그 (+ - * / % 등은 내부적으로 플래그로 지정) 에 따라 연산
data와 Return에 해당 값을 저장

print는 우측 블록에 따라서 숫자 or 텍스트 값을 받고 출력 버퍼에 푸시함

Num과 Str 쪽은 이름을 데이터로 바꾸는 것으로 끝임

Var는 우측의 값을 저장함

Var 포인터는 Var의 값을 우측의 값으로 저장하고 Var의 값을 리턴함

이정도임

결론: 본인이 직접 짠 코드는 수천줄이 되어도 쉽게 이해를 하는 것 같고 그 수혜를 이번 프로젝트에서 가장 많이 받았다고 보면 된다고 보고, 이번 프로젝트에서는 계획과 시간 활용이 매우 아쉬웠던 것 같음

만약에 협업을 하는 환경이었다면 이러한 코드를 짜는 것은 상당히 나쁜 일이라고 생각을 하고 그러니 다음에 긴 코드를 작성할때는 확실한 계획과 시간 활용이 무척이나 중요한 일이라고 생각하게 되었음

출처: 나, SDL 라이브러리에 대한 검색은 퍼플렉시티 pro가 맡았음, SQLite 부분은 퍼플렉시티 pro가 맡았다고 봐도 무방함

출처 2: 20240954 장용호

해당 프로젝트의 깃허브 링크:

https://github.com/jang121314/lab24_2_1/tree/main/%EB%B8%94%EB%A1%9D%20%EC%BD%94%EB%94%A9%20%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8%20%ED%8C%8C%EC%9D%BC

발표에 쓴 파일 드라이브 링크: (완성본은 저장 안해서 이 본은 미완성 본임)

https://drive.google.com/file/d/10LcM9TWlSshXCOcqjBO_QA8cxAh9S8U5/view?usp=drive_link

재 완성한 본 드라이브 링크: (사실 변수 포인터 색변만 구현함)

https://drive.google.com/file/d/1ImMxOSNt7IbgtnHqKJ-LHaaqKJwSfRRa/view?usp=sharing

그리고 마지막으로: 수고하셨습니다.