2022 CAT-CERT 스터디 보고서

컴퓨터정보공학부 202221168 김지혜

2022.05.16 C\_study-7

PRINT(), PRINTINFO(), swap() 함수 구현

소스코드

|  |
| --- |
| #include <stdio.h>  typedef struct student {  char Name[100];  int age;  int digit;  }Student;  void swap(Student\* a, Student\* b) {  Student tmp = \*a;  \*a = \*b;  \*b = tmp;  }  void PRINT(const Student x) {  Student info =x;  printf("Name : %s \n", info.Name);  printf("age : %d \n", info.age);  printf("digit : %d \n", info.digit);  }  void PRINTINFO(const Student\* x) {  Student \*ptr = x;  printf("Name : %s \n", ptr->Name);  printf("age : %d \n", ptr->age);  printf("digit : %d \n", ptr->digit);  }  int main() {  Student a = { "Miru", 3, 191128 };  Student b = { "Ruri",3, 200201 };  PRINT(a);  printf("\n");  PRINT(b);  printf("\n");  swap(&a, &b);  PRINTINFO(&a);  printf("\n");  PRINTINFO(&b);  } |

출력 결과

텍스트이(가) 표시된 사진

자동 생성된 설명

#실습 과제 1

친구관리프로그램 만들기

소스코드

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  typedef struct \_friend {  char\* name;  char\* message;  int status;  }friend;  friend \*list[100];  int index;  int Addfriend();  int Delfriend();  int Viewfriend();  int Allfriend();  int main() {  int num = 0;  while (1) {  friend\* frd = 0;  printf("==========Friend Menu==========\n");  printf("\t1. Add friend\n");  printf("\t2. Delete friend\n");  printf("\t3. View friend\n");  printf("\t4. All friend\n");  printf("\t5. Exit\n");  printf("===============================\n");  printf("Choose menu:");  scanf("%d", &num);  if (num == 1)  Addfriend();  else if (num == 2)  Delfriend();  else if (num == 3)  Viewfriend();  else if (num == 4)  Allfriend();  else {  exit(0);  }  }    }  int Addfriend(){  int size = 0;  int msize = 0;  friend\* frd = 0;  int len;  len = strlen(list);  frd = (friend \*)malloc(sizeof(friend)); //friend자료형크기만큼 할당  printf("name size:");  scanf("%d", &size);  frd->name = (char\*)malloc(size); //name사이즈만큼 할당  printf("name: ");  scanf("%s", (\*frd).name);    printf("message size:");  scanf("%d", &msize);  frd->message = (char\*)malloc(msize);  printf("message :");  scanf("%s", (\*frd).message);  printf("status (put 0(:offline) or 1(:away) or 2(:online):");  scanf("%d", &(\*frd).status);  printf("\n");  list[index] = frd;  index++;  }  int Delfriend() {  printf("삭제할 친구의 인덱스번호를 적어주세요!:");  scanf("%d", &index);  free(list[index]);  list[index] = NULL;  printf("삭제되었습니다\n");  printf("\n");  }  int Viewfriend() {  char search[100];  printf("검색할 친구의 이름을 적어주세요! : ");  scanf("%s", &search);    for (int i = 0; list[i] != '\x00'; i++) {  if (list[i]->name != NULL) {  if (strcmp(search, list[i]->name) == 0) {  printf("MY FRIEND!!\n");  printf("name : %s\n", (\*list[i]).name);  printf("message : %s\n", (\*list[i]).message);  if ((\*list[i]).status == 0)  printf("status : offline\n");  if ((\*list[i]).status == 1)  printf("status : away\n");  if ((\*list[i]).status == 2)  printf("status : online\n");  printf("\n");  }  }  }  }  int Allfriend() {  for (int i = 0; i<100; i++) {  if (list[i] != NULL) {  printf("name : %s\n", list[i]->name);  printf("message : %s\n", list[i]->message);  if ((\*list[i]).status == 0)  printf("status : offline\n");  if ((\*list[i]).status == 1)  printf("status : away\n");  if ((\*list[i]).status == 2)  printf("status : online\n");  printf("\n");  }  }  } |

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

조사 과제

#1 calloc, realloc 조사하고 실습

**Calloc 함수의 원형**

void\* calloc(size\_t elt\_count, size\_t elt\_size)

* elt\_size 크기의 변수를 elt\_count 개 만큼 저장할 수 있는 메모리 공간을 할당하라는 의미를 갖음

**malloc함수와 calloc의 차이점**

:malloc은 할당된 공간의 값을 바꾸지 않지만 (쓰레기 값이 그대로 들어있음) calloc은 할당된 공간의 값을 모두0으로 바꾼다

* 배열을 할당하고 모두 0으로 초기할 필요가 있을 경우 calloc을 사용한다.

소스코드

#include<stdio.h>

#include<stdlib.h>

int main() {

int\* p = (int\*)calloc(1, sizeof(int));

printf("초기 : %d\n", \*p);

\*p = 20;

printf("간접 연산을 수행한 후: %d\n", \*p);

free(p);

return 0;

}

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

**Realloc 함수** : 할당한 공간의 크기를 바꿀 때 realloc함수를 사용함

realloc함수의 원형

void\* realloc(void\* p, size\_t size);

* 이미 할당한 포인터 변수를 p에 넣고 바꾸고 싶은 공간의 크기를 size에 입력한다
* \*여기서 void는 어떤 타입이든 포인터를 알려달라는 뜻임

소스코드

#include<stdio.h>

#include<stdlib.h>

int main() {

int arr1[5];

int\* arr2;

int i;

for (i = 0; i < 5; i++) {

arr1[i] = i + 1;

}

arr2 = (int\*)malloc(sizeof(int) \* 5); //20

for (i = 0; i < 5; i++) {

arr2[i] = arr1[i];

printf("%d", arr2[i]);

}

printf("\n");

realloc(arr2, sizeof(int) \* 7); //28

for (i = 0; i < 7; i++) {

arr2[i] = arr1[i];

printf("%d", arr2[i]);

}

free(arr2);

return 0;

}

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

#2 공용체(union)에 대해 조사 & 간단한 실습

공용체 : 서로다른 자료형을 가진 데이터들이 동일한 기억장소를 서로 공유하며 기억되는 형태

(어느 한 시점에서는 기억 장소를 오직 하나의 데이터 형만 기억함)

* 메모리를 절약하고자 할 때 여러 자료형의 데이터를 하나의 기억장소에 넣고 필요한 작업을 하고자 할 때 주로 사용됨

공용체의 일반 형식

|  |
| --- |
| Union 공용체 이름 {  자료형 멤버이름;  }; |

공용체는 보통 main함수 바깥에 정의함 ; 함수안에서 정의하면 해당함수 안에서만 사용가능

공용체의 전체 크기는 가장큰 자료형을 기준으로 정해짐

소스코드

#include<stdio.h>

typedef union box{

int a;

int b;

double c;

} Data;

int main(void)

{

Data box;

box.a = 14;

printf("%x\n", box.b);

box.c = 6.15;

printf("%d\n", box.a);

printf("%d\n", box.b);

printf("%g\n", box.c);

return 0;

}

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

#3 열거형(enum)에 대해 조사

프로그래밍 도중 자주 사용하는 변수값 을 정수로 표현하여 사용함

열거형을 사용하면 변수가 갖는 값에 의미를 부여할 수 있어 프로그램 가독성이 향상 됨

정의 선언;

|  |
| --- |
| “enum” “Tag”{ 내용 들 }; |

-> 구조체나 공용체는 중괄호 안에 변수를 선언하지만 열거형은 콤마로 구분자를 넣어서 사용할 문자를 지정한다

\*열거형도 정수 0부터 시작한다.(대신 자신이 원하는 값으로 지정할 수 있음)

소스코드

|  |
| --- |
| #include<stdio.h>  enum fourseason {  spring = 1,  summer,  fall,  winter  };  int main() {  printf("spring : %d\n", spring);  printf("summer : %d\n", summer);  printf("fall : %d\n", fall);  printf("winter : %d\n", winter);  } |

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

#4 Socket의 개념(종류, 역할, 용도 등 자세히)과

Socket 통신(TCP 프로토콜 통신)의 과정 조사

소켓은 프로세스가 드넓은 네트워크 세계로 데이터를 내보내거나 혹은 그 세계로부터 데이터를 받기 위한 실제적인 창구 역할을 함. 그러므로 프로세스가 데이터를 보내거나 받기 위해서는 반드시 소켓을 열어서 소켓에 데이터를 써 보내거나 소켓으로부터 데이터를 읽어들인다.

떨어져 있는 두 호스트를 (서버와 클라이언트가 포트를 통해 실시간으로 양방향 통신을 함) 연결해주는 도구로 인터페이스의 역할을 하는데 데이터를 주고 받을 수 있는 구조체로 소켓을 통해 데이터 통로가 만들어짐

소켓 => 프로토콜, IP주소, 포트넘버

**SOCKET 통신의 종류:**

|  |  |
| --- | --- |
| **스트림(TCP)** | **데이터그램(UDP)** |
| -양방향으로 바이트 스트림을 전송, 연결 지향성  -오류 수정, 전송처리, 흐름제어 보장  -송신된 순서에 따라 중복되지 않게 데이터를 수신함  -소량의 데이터보다 대량의 데이터 전송에 적합함 | -비연결형소켓  -데이터의 크기에 제한이 있음  -확실하게 전달이 보장되지 않음, 데이터가 손실돼도 오류가 발생하지 않음  -실시간 멀티미디어 정보를 처리하기 이해 주로 사용함 |

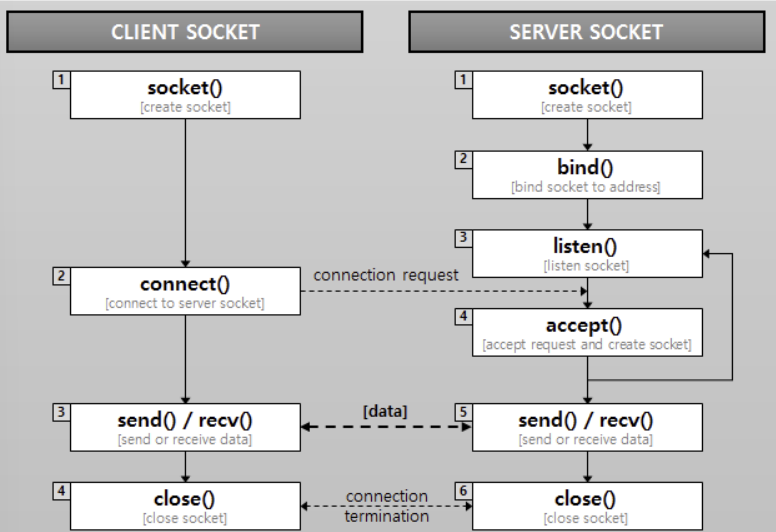
**SOCKET 통신의 특징**

-SERVER와 CLIENT가 계속 연결을 유지하는 양방향 통신

-SERVER CLIENT가 실시간으로 데이터를 주고받는 상황이 필요한 경우에 사용됨

-실시간 동영상 스트리밍이나 온라인 게임등과 같은 경우에 자주 사용됨

**소켓 통신 흐름**



|  |  |
| --- | --- |
| 클라이언트 소켓   1. Socket():소켓을 생성 2. Connect(): 통신할 서버에 설정된 ip와 포트 번호에 통신을 시도 3. 통신 시도 시, 서버가 accept()함수로 클라이언트의 socket descriptor를 반환함 4. 이를 통해 클라이언트와 서버는 서로 send(), recv()하며 데이터를 송수신함 | 서버 소켓   1. Socket(): 클라이언트와 서버를 연결한 연결용 소켓을 생성 2. Bind() : ip와 포트번호를 결합 3. Listen(): 클라이언트의 접근 요청을 수신하는 대기열을 만들어 몇 개의 클라이언트를 대기 시킬지 결정함 4. 클라이언트가 connect()해오기 전까지 대기하기 때문에 listen()은 별도 스레드를 만들어서 처리함 5. Accept() 새로운 소켓을 만들어 클라이언트와 실질적인 연결 수행 |

#5 why?

Q. char는 1byte, int는 4byte 이어서 값이 5가나와야하는데 왜 값이 8일까?

텍스트이(가) 표시된 사진

자동 생성된 설명

-> 구조체의 크기는 가장 큰 사이즈에 맞춰서 메모리가 할당되기 때문이다. 그래서 int의 사이즈는 4인데 char의 사이즈는 1이기 때문에 사이즈가 int보다 작으므로 4의 메모리를 사용한다.

4+4=8로 메모리의 공간은 8이 나온다.