시작하기 전.

컴퓨터 메모리의 공간은 스택, 힙, 데이터 부분으로 나눌 수 있습니다.

첫번째로, 데이터 영역은 전역 변수와 static 변수가 할당되는 영역으로 프로그램 시작과 동시에 할당되고 프로그램이 종료되어야 메모리에서 소실됩니다

두번째로, 스택 영역은 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역으로 함수 호출이 완료되면 사라집니다.

|  |  |
| --- | --- |
| #include<stdio.h>  void stk1(int);  void stk2(int);  int a = 10; // 데이터  int b = 20; // 영역에 할당  int main() {  int i; // i가 스택 영역에 할당  stk1(i);  stk2(i);  return 0;  }  void stk1(int c){ // c와 d가  Int d = 30; // 스택 영역에 할당 됨  }  void stk2(int e){ // e와 f가  int f = 40 // 스택 영역에 할당됨  } | Int 형 변수 a, b는 main 함수가 호출 되기 전 데이터 영역에 할당되고 프로그램 종료까지 메모리 상에 존재  main 함수 호출 시 int형 변수 i는 지역변수로서 스택영역에 할당 됨.  skt1 함수 호출 시 c와 d를 스택영역에 할당.  skt1 함수 호출이 끝나면 스택영역에서 삭제.  skt2 함수 호출 시 e와 f를 스택영역에 할당.  skt2 함수 호출이 끝나면 스택영역에서 삭제. |

세번째로, 힙 영역은 필요에 의해 동적으로 메모리를 할당할 때 사용합니다.

|  |  |
| --- | --- |
| Int main() {  Int i = 10;  Int arr[i];  Return 0;  } | 컴파일을 하는 동안 i가 4바이트 라는 것은 알 수 있으나 그 값이 10으로 초기화 했다는 사실은 무시하고 지나가기 때문에, 컴파일러는 이 값이 얼마가 될지 알 수 없다.  (10으로 초기화 했다는 사실은 실행되는 동안 결정된다.) |

그렇다고 무한정 정적 할당하는 것은 메모리를 지나치게 낭비하게 됩니다.

따라서 스택과 데이터 영역만으로는 프로그램이 실행되는 동안 할당해야하는 메모리의 크기를 결정 할 수 없기에. 이처럼 사용자 요구에 맞게 메모리를 할당해 주기 위해서 메모리 동적 할당을 사용하여 힙 영역에 메모리를 할당하여야 합니다.

동적 할당이란 간단하게 말해서 필요할 때 필요한 만큼 메모리를 할당하여 사용하고, 다 쓰면 버리는 것이라고 할 수 있습니다.

독점적인 메모리 영역을 차지하고 있는데, 일단 값을 기억할 수 있지만 이름이 없어 오로지 포인터로만 접근할 수 있습니다. 그래서 뒤에서 설명할 malloc 함수가 리턴 하는 포인터는 반드시 적절한 타입의 포인터 변수로 대입 받아야 합니다.

C 프로그래밍에서는 malloc() 과 calloc() 함수 등을 사용하여 메모리를 동적으로 생성하며, free() 함수를 사용해 동적으로 할당된 메모리를 명시적으로 해제합니다.

C++ 프로그래밍에서는 이러한 함수들을 더 편하게 사용할 수 있도록 new아 delete 연산자를 사용하여 할당 및 해제합니다.

1. void \*malloc(size\_t size);

malloc은 프로그램이 필요로 하는 양만큼 운영체제에게 할당을 요청하며 운영체제는 사용하지 않는 빈 영역(힙)을 찾아 요청한 만큼 메모리를 할당하며 그 시작 번지를 리턴 합니다.

할당된 메모리를 어떠한 목적으로 사용하는지 알 수 없기에 malloc은 void \* 형으로 리턴하며 받는 쪽에서 원하는 타입으로 캐스팅 해야합니다.

10 바이트가 필요하면 malloc(10), 1000바이트가 필요하면 malloc(1000)이라고 호출하면 됩니다. 실행 중 할당하는 것이므로 malloc(Num)과 같이 변수도 사용할 수 있습니다.

Ex) Int \*I = (int \*) malloc(sizeof(int)); 에서

3 2 1

1에서 Sizeof(int) 는 4라는 값을 전달하면서 malloc을 호출한다.

할당된 메모리의 주소가 void\*형으로 리턴되고, 리턴되는 void\*를 2에서 int\*형으로 변환시킨다.

변환시킨 int\*형을 변수 i에 대입한다.

1. void free(void \*memblock);

free 함수는 동적으로 할당한 메모리를 해제합니다. 메모리를 다 사용한 프로그램은 반드시 **free 함수를 통해 메모리를 해제**하여야 합니다. 할당만 하고 free 하지 않으면 이 메모리는 시스템을 재부팅 하기 전에는 다른 프로그램이 사용할 수 없습니다. 그러므로 꼭 free를 하여 메모리를 해제 하여야 합니다.

int \*dm; // 시작 번지를 기억해야 하므로 정수형 포인터 dm 선언

ar = (int \*)malloc(10\*sizeof(int)); // 메모리 양을 바이트 단위로 전달하기 때문에,

. . . // sizeof(int)에 10을 곱함.(\*반드시 바이트 계산에 sizeof 이용)

free(dm); // \*\*\*\*free 함수로 메모리 해제\*\*\*\*

1. void \*calloc(size\_t num, size\_t size);

malloc과 마찬가지로 메모리를 할당하지만, 필요한 메모리를 지정하는 방식이 다릅니다.

malloc과 다르게 calloc은 메모리를 두개의 값으로 나누어 전달받습니다.

간단하게 말하면 malloc은 몇 바이트 할당 해주세요 요청하는 것이고 calloc은 몇 바이트짜리 몇 개 할당해 주세요 라고 요청하는 것 입니다.

ar = (int \*)**malloc**(10\*sizeof(int));

ar = (int \*)**calloc**(10, sizeof(int));

위 두 호출문은 서로 동일합니다. 하지만 calloc은 메모리 할당 후 전부 0으로 초기화 한다는 점이 malloc과 차이가 있습니다. 할당 후 바로 배열을 초기화해야 한다면 malloc보다는 calloc을 이용하여 할당하는 것이 더 편리합니다.

1. void \*realloc(void \*memblock, size\_t size);

realloc은 이미 할당된 메모리의 크기를 바꾸어 재할당을 할 시에 사용합니다. 최초 할당 크기보다 더 크거나 작게 메모리를 재 할당하는 것이 가능하나, 보통은 확대 재할당에 이용합니다.

첫번째 인수가 NULL일 경우, 할당되어 있지 않은 경우에는 새로 메모리를 할당하므로 realloc의 동작은 malloc과 같아집니다. 또한 size가 0일 경우에는 할당을 취소하라는 이야기이므로 free와 같아집니다.

Void main(){

int \*ar;

ar= (int \*)malloc(5\*sizeof(int)); //ar을 20 바이트 크기로 할당

ar[4]=1234;

ar=(int \*)realloc(ar, 10\*sizeof(int)); //ar을 40 바이트 크기로 확장

ar[9]=5678;

printf(“ar[4]=%d, ar[9]=%d\n”, ar[4],ar[9]); // 실행 결과 ar[4]=1234, ar[9]=5678 출력.

free(ar);

}

C언어의 기본적인 메모리 할당, 해제 함수는 malloc, free 입니다. 이에 비해 C++에서는 malloc, free 대신에 사용할 수 있는 new, delete라는 할당 연산자를 따로 제공하는데, 이 연산자들은 객체의 생성자와 파괴자를 호출하기도 합니다.

C++에서 사용하는 new와 delete 연산자는 malloc, free 연산자와는 다르게 크기를 바이트 단위로 계산하는 일을 거치지 않아도 됩니다.

New 연산자의 사용법을 간단히 말하자면 아래와 같습니다.

‘주소를 저장할 포인터’ = new ‘할당하고 싶은 크기의 자료형’;

Int \*p = new int;

Int \*ex = new int(10)

C언어에서 malloc 함수를 사용하여 40바이트 메모리를 할당하려면

Int \*p = (int \*)malloc(10\*sizeof(int));

이와 같이 코드를 구성하면 되었는데,

C++에서 new 연산자를 사용하면 아래와 같이 간단히 할 수 있습니다.

Int \*p = new int[10];

자신이 원하는 자료형을 new 뒤에 그대로 적으면 됩니다.

delete 연산자는 동적 할당 된 메모리를 해제할 수 있습니다. 보통 new 연산자로 동적 할당을 하였다면 delete 연산자를 이용하여 할당된 메모리를 해제합니다.

단, **단순하게 메모리가 할당 되었는지. 그룹으로 메모리가 할당되었는지** 꼭 구분하여 사용해야 합니다.

Int \*p = new int(10) int \*p = new int[10]

. . . . . .

delete p; delete p[];

c++에서는 객체라는 새로운 자료형이 추가되어 단순한 함수형이라 사용할 수 없는 malloc 대신 객체에도 동적 할당 할 수 있는 new 연산자를 사용합니다.

Class Study{

Private:

Int m\_data;

Public:

Study()

{

m\_data = 0;

}

};

이 객체를 malloc을 이용하여 메모리를 할당하면 아래와 같습니다.

Study \*p = (Study \*)malloc(sizeof(Study));

객체는 선언이 되면 메모리만 할당되는 것이 아니라 객체 생성자가 자동으로 호출되어 객체 스스로 초기화하는 작업을 진행하는데, malloc을 사용하면 단순히 메모리만 할당되고 객체 생성자는 호출되지 않게 됩니다. (m\_data를 위한 메모리는 만들어지지만, 0을 대입하는 객체 생성자는 호출되지 않습니다.) 따라서 아래와 같이 new 연산자를 사용합니다.

Study \*p = new Study;

만약 기본 객체 생성자가 아닌 정수 값을 하나 사용하는 객체 생성자를 호출하고 싶으면 아래와 같습니다.

Study \*p = new Study(5);

그리고 이렇게 할당 된 메모리를 해제하고 싶으면 delete 연산자를 사용하면 됩니다. Delete 연산자를 사용하면 메모리를 해제하기 전에 자동으로 객체 파괴자가 호출되고 메모리가 해제됩니다.

문제 1.

malloc을 이용하여 프로그램 실행 직후 입력 받은 n일동안 하루에 벌어들인 수입을 입력하여 처리할 수 있게 하시오.

1. 몇일 일하는지 입력 받을 것.
2. 각각 하루의 수입을 입력할 것.
3. N일 동안의 수입의 총 합을 구할 것.

문제 2. 임의의 n개의 숫자를 입력 받아 그중 가장 큰 숫자를 출력하는 코드를 설계하시오.

1. 비교할 수들의 전체 개수를 입력 받을 것.
2. 전체 입력 받은 수의 개수만큼의 임의의 수를 입력할 것.
3. 입력 받은 수 중 가장 큰 수가 출력되도록 할 것.

문제3. 임의의 개수의 수를 입력 받아 버블 정렬하는 코드를 설계하시오.

1. 비교할 수들의 전체 개수를 입력 받을 것.
2. 전체 입력 받은 수의 개수만큼의 임의의 수를 입력할 것.
3. 입력 받은 수들을 버블 정렬하여 출력할 것.

문제 1 답

#include <stdio.h>

#include <stdlib.h>

void main(){

int \*account;

int i;

int days;

int sum;

printf("일한 날짜를 입력하시오 :");

scanf("%d", &days);

account = (int\*)malloc(days\*sizeof(int));

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

printf("%d일의 수입을 입력하시오", i+1);

scanf("%d", &account[i]);

}

sum = 0;

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

sum += account[i];

}

printf("%d일을 일하고, %d원을 벌었습니다.", days,sum);

free(account);

}

문제 2답

#include <stdio.h>

#include <stdlib.h>

void main(){

int \*count;

int i;

int num;

int temp;

printf("개수를 입력하시오");

scanf("%d", &num);

count = (int\*)malloc(num\*sizeof(int));

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

printf("숫자를 입력하시오");

scanf("%d", &count[i]);

}

temp = 0;

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

if(count[i] > temp){

temp = count[i];

}

}

printf("가장 큰 숫자는 %d입니다.", temp);

free(count);

}

문제 3답

#include <stdio.h>

#include <stdlib.h>

int main(){

int \*count;

int i = 0;

int num;

int temp;

int a;

printf("개수를 입력하시오");

scanf("%d", &num);

count = (int\*)malloc(num\*sizeof(int));

for(a=0; a<num; a++){

printf("숫자를 입력하시오");

scanf("%d", &count[a]);

}

do{

int j = 0;

do{

if(count[j]>count[j+1]){

temp = count[j];

count[j] = count[j+1];

count[j+1] = temp;

}

j++;

}while(j<num-1-i);

i++;

}while(i<num-1);

for(int i = 0; i<num; i++){

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

}

free(count);

return 0;

}