DINAMIČNA ALOKACIJA POMNILNIKA

Življenska doba lokalnih spremenljivk je enaka trajanju funcije v kateri so deklarirane, globalne spremenljivke pa imajo življensko dobo enako trajanju programa. Lokalne spremenljivke se nahajajo v skladu (stack).

Dinamične spremenljivke so spremenljivke, za katere se pomnilnik (kopica ali heap ja del pomnilnika, katerega kontrolira OS) rezervira, kadar to zahtevamo.

Dinamična spremenljivka se mora pobrisati iz programa podobno, kot se je ustvarila!!

Dinamične stukture je smiselno uporabljati, ko uporabljamo:

* obsežne podatke,
* podatke, katerih velikost ni znana v času prevajanja ( datoteke, nizi,...)
* objekte pri katerih želimo kontrolirati njihovo življensko dobo,
* veliko podatkov in nam zmanjkuje sklada,...

**C:**

S funkcijo **malloc()** zahtevamo dodatni pomnilnik.

Funkcija **free()** izbriše dinamično spremenljivko oz. „sprosti“ rezervirani del pomnilnika.

Uporabljamo tudi funkcije realloc(), calloc(), ...

.

Primer:

#include<stdio.h>

#include<stdlib.h> // namesto alloc.h je lahko tudi stdlib.h

struct oseba

{

char ime[20];

char priimek[30];

float visina;

};

void main()

{

struct oseba \*ptr;

int j;

printf("sizeof(struct oseba)=%d\n",sizeof(struct oseba)); //izpiše velikost strukture

for (j=0;j<4;j++)

{

ptr=(struct oseba \*) malloc(sizeof(struct oseba)); //vrne naslov na katerega kaze ptr

printf("ptr=%d \n",ptr);

}

free(ptr);

}

Komentar:

(struct oseba \*) je tipa cast

Cast operator se uporablja pri konverzijah podatkovnih tipov:

npr. (double)x, če je x znak

**C++:**

Operator **new** alocira (rezervira) dodatni (nov) pomnilnik

Operator **delete** izbriše dinamično spremenljivko oz. „sprosti“ rezervirani del pomnilnika.

#include <iostream>

using namespace std;

struct oseba

{

char ime[20];

char priimek[30];

float visina;

};

int main()

{

struct oseba \*ptr;

int j;

cout<< sizeof(struct oseba); //izpiše velikost strukture

for (j=0;j<4;j++)

{

ptr=new struct oseba; //vrne naslov na katerega kaze ptr

cout<<"ptr = " <<ptr<<endl;

}

delete ptr;

return 0;

}