Семинар по „Увод в програмирането“

Стекова и динамична памет

1. Stack memory

Стековата памет наподобява структурата от данни **Stack** по заделянето и изтриването на памет. Всички променливи, за които е заделена памет в някаква функция, техния цикъл на живот приключва в края на scope-а на тази функция. С други думи паметта, заделена в дадена функция, може да се достъпва само в нея до края на изпълнението на тази функция. Реда, в който се заделя памет, е обратен на реда, в който се освбожда. Паметта се заделя **преди** да се компилира програмата. Стековата памет е по-малка от тази на хийпа(Heap memory), но пък по-бърза. Това е така поради начина на заделяне и изтриване на паметта, което наподобява работата на стек.

Ползи на стековата памет:

- Автоматично алокира и деалокира памет
- По-бърза от хийпа(Heap memory)
- Не ангажира програмиста да мисли за нейното освобождаване

Пример в сравнение със статичната памет:

```
1   #include <iostream>
2
3   void foo()
4   {
5       int num = 0;
6       std::cout << num << std::endl;
7       num++;
8   }
9
10  int main()
11  {
12      foo();
13      foo();
14      foo();
15      foo();
16
17      return 0;
18  }
```

Резултат:

2. Heap memory

Хийп паметта няма нищо общо със структурата от данни **Heap**, както при стековата памет. Тук под хийп памет си има предвид купчина. Това е така, защото разположението на паметта в хийпа е хаотично. Тоест не е нужно паметта да е последователно разположена, както при стека. Когато заделяме памет динамично, в хийп паметта, програмата започва да търси за свободно парче памет, което отговаря на изискванията за размер на тази памет и я запазва като връща указател към нея. Заделянето на памет в хийп паметта е „бавна операция“ в сравнение със заделянето в стековата памет, защото заделянето на стекова памет е просто една инструкция на процесора. За разлика от стековата памет, тук ние като програмисти трябва да се грижим веднъж като заделим блок от памет, той да бъде изтрит до края на изпълнението на програмата. В противен случай ще имаме изтичане на памет(**memory leaks**). Данните заделени в хийп паметта могат да бъдат достъпвани навсякъде, стига да имаме указатели, които да сочат към тях. Хийп паметта е по-голяма от стековата памет.

Заделянето на памет в хийпа или още динамичното заделяне на памет се случва с ключовата дума **new.** Това, което **new** прави е, че надниква в хийп паметта, намира свободен блок от памет и ни връща указател към него. Съответно, за да освободим тази памет след това ни е нужна ключовата дума **delete.**

Пример:

```cpp
#include <iostream>

int main()
{
    //variable in stack memory
    int var = 10;

    //variable int heap memory
    int* dynamicVar = new int(10);
    //another way is
    //int* dynamicVar = new int;
    //*dynamicVar = 10;
    std::cout << *dynamicVar;

    //array in stack memory
    int arr[3];
    arr[0] = 1;
    arr[1] = 2;
    arr[2] = 3;

    //array in heap

    int* dynamicArr = new int[3];
    dynamicArr[0] = 1;
    dynamicArr[1] = 2;
    dynamicArr[2] = 3;

    delete dynamicVar;
    delete[] dynamicArr;

    return 0;
}
```

Когато изтриваме динамично заделен масив, трябва да добавим едни **[]** скоби след **delete.**

Друг пример:

```cpp
#include <iostream>

int* foo()
{
    int* var = new int(0);
    *var = 12;
    return var;
}

int main()
{
    int* ptr = foo();

    std::cout << *ptr;

    delete ptr;

    return 0;
}
```