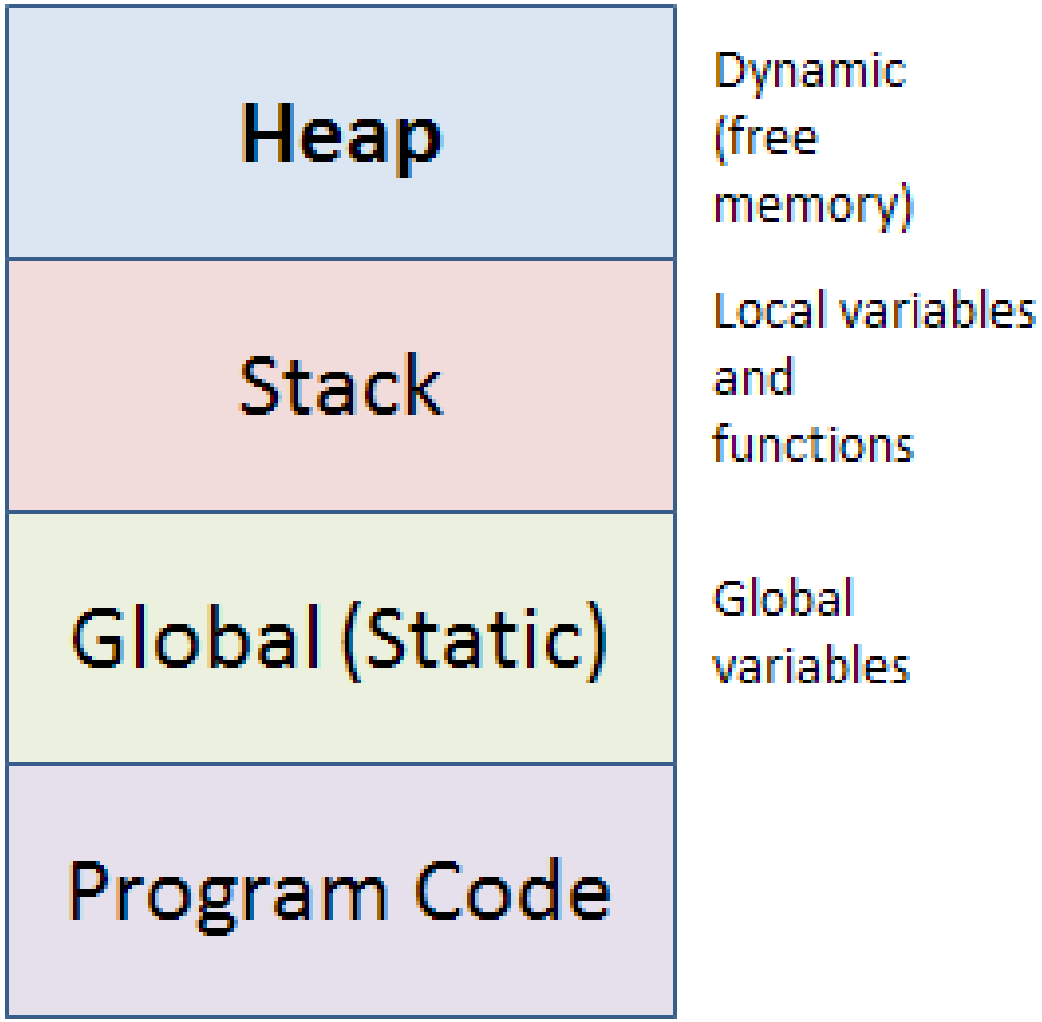
Видове памет



Статични данни (глобални)

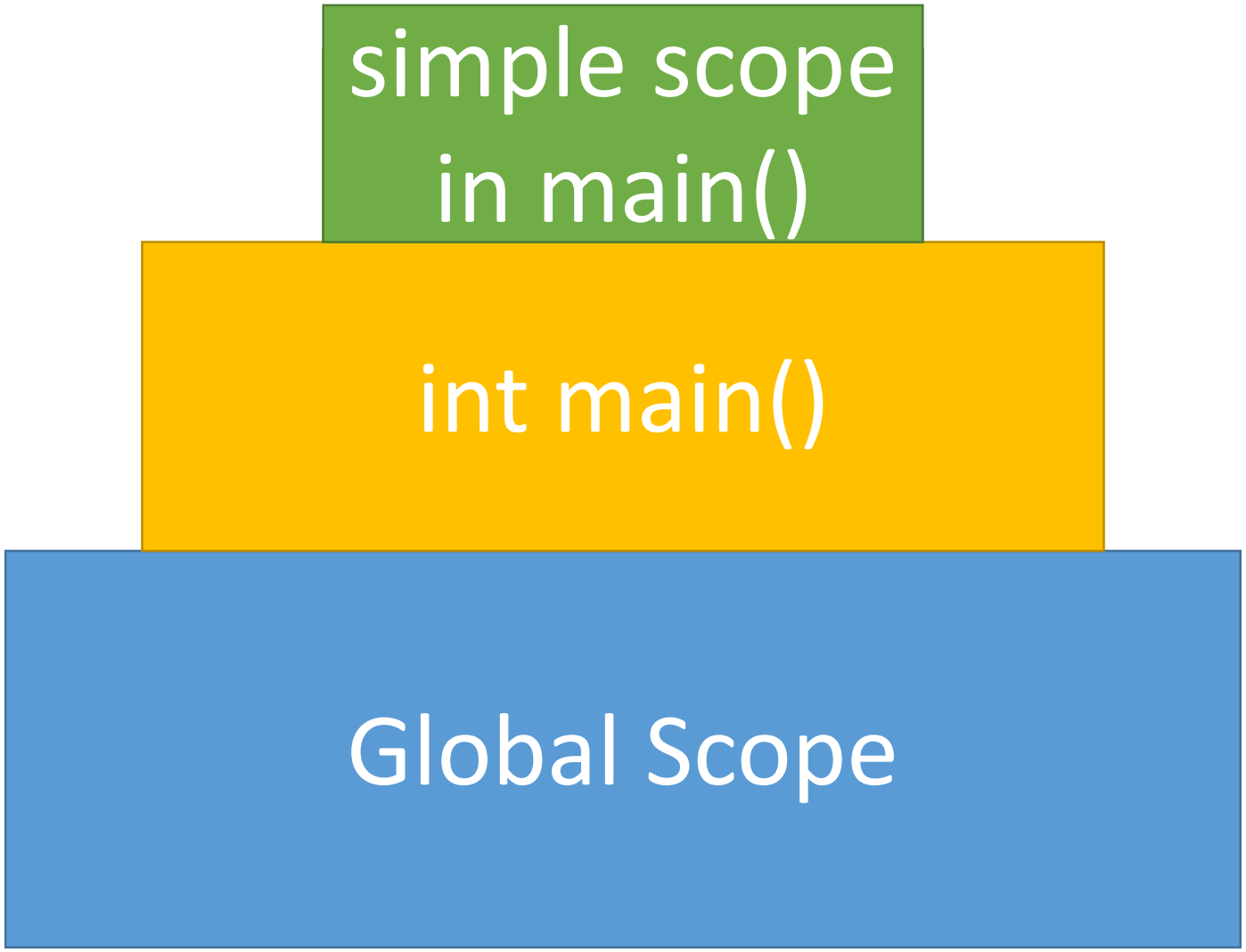
- Данни, които съществуват до края на програмата
- Досега знаете за глобални променливи
- Добра практика е глобалните променливи да са константни

По-точно обяснение на scope

- Дотук за scope знаем, че:
  - Пази данните на всички променливи на scope-а, в който се намира
  - Ако се създаде нова променлива със запазено име от външен scope, то се забравя за старата променлива за този scope
  - Когато свърши даден scope, всички данни, създадени в него, изчезват

- Освен това:
  - данните, които заделяме в scope, използват така наречената стекова памет
  - както сте се сетили, всички функции представляват scope, включително и main

Стек – Купчина



Стек in a nutshell

- Стек е структура от данни, която няма да разглеждаме сега
- Накратко, в него се вкарват данни и единственият начин да се изкарат данни е да се изваждат отзад напред вкараните данни
- Пример:
  - Голяма колона коли засяда в тясна улица без изход
  - Единствено последната кола може да излезе на заден ход
  - След това само предпоследната може да излезе на заден ход
  - Така след краен брой стъпки и последната ще излезе

Дефиниция за стек според fmi.wiki

Стек

- FILO: First-In-Last-Out
- *Принцип на библията: Последните ще бъдат първи*

Стековата памет на интуитивно ниво

- Майстор Тричко прави ремонт. Задачата му е да смени кранчето за студената вода.

1.Той започва да го сменя, но се обляга на мивката и я изкъртва.

2.Сега задачата му е първо да смени мивката, но докато го прави спуква тръба.

3.Сега задачата му е да оправи тръбата, но за да го направи трябва първо да спре течащата вода.

4.Той спира водата.

3.След това оправя тръбата.

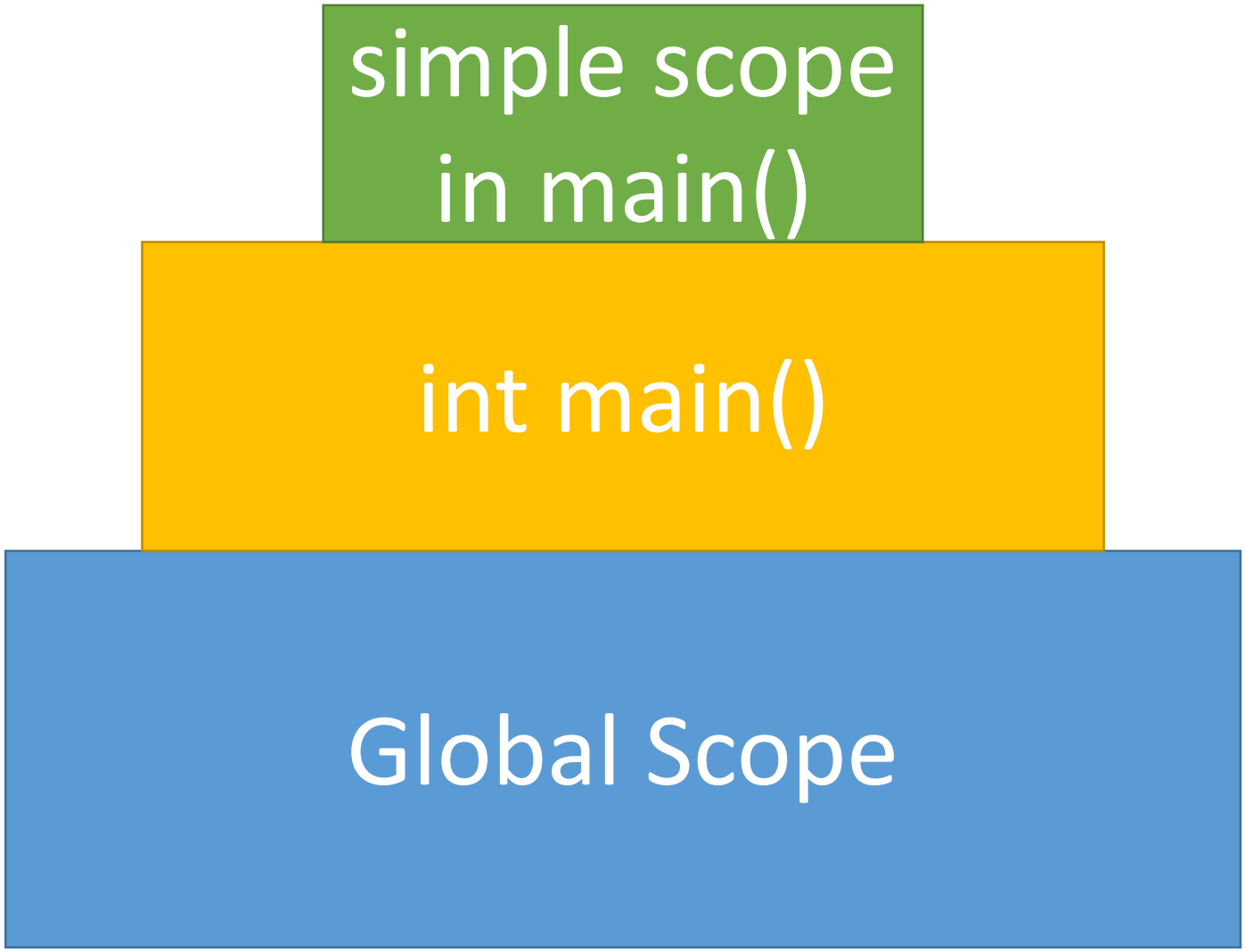
2. После оправя мивката.

1.Накрая сменя и кранчето за студената вода.

Стек – Купчина

```
int a = 5;
int main()
{
        char a = ‘Q’;
        {
                double a = true;
        }
}
```

Стек – Купчина



Стек – Купчина

```
int a = 5;
int main()
{
        char a = ‘Q’;
        {
                double a = true;
        }
}
```

Стекова памет

- Заделя се в момента на дефиниция
- Всеки scope знае каква стекова памет е заделил
- При край на scope, той освобождава всичката стекова памет, която е заделил
- Последно заделената стекова памет се освобождава първа
- От горното свойство идва и наименованието на този вид памет

Стекова памет

- Програмистът няма контрол над управлението на паметта
- Стекова памет не може да се освободи по-рано (преди края на блока)
- Стекова памет не може да се запази за по-дълго (след края на блока)
- До голяма степен работата със стековата памет е предопределена преди началото на програмата

Статична срещу стекова памет

- И двете имат имена на заделената памет
- При статичната памет, данните съществуват до края на програмата, докато при стековата, до края на scope-а, в който са били създадени

Какво имаме досега

- Имаме данни, които се запазват по време на компилация
- Тези данни си имат имена
- До голяма степен сме ограничени от езика да извършваме наглед прости операции, като:
  - контрол над паметта, която използваме, в реално време
    - заделяне на памет по време на изпълнение на програмата
    - освобождаване на памет по време на изпълнение на програмата
- Пример:

```
int n;
std::cin>>n;
int arr[n];
```

Динамична памет (heap)

- Може да бъде заделена и освободена по всяко време на изпълнение на програмата
- Областта за динамична памет е набор от свободни блокове памет
- Програмата може да заяви блок с произволна големина
- За нейното управление се грижи операционната система
- Съответно не представлява някаква променлива, към която можем да се обърнем по име