«ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՈՒՆՔՆԵՐ» դասընթաց

այլ ոլորտներից դեպի տեխնոլոգիական ոլորտ սկսնակների համար

edu2020.am

ԴԱՍ #12

ՀԱՅ-ՌՈՒՍԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ
ԲԱՐՁՐ ՏԵԽՆՈԼՈԳԻԱԿԱՆ
ԱՐԴՅՈՒՆԱԲԵՐՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

Ռեկուրսիվ ֆունկցիաներ

- Ֆունկցիան հրամանների հաջորդականություն է, որը կարելի է կանչել ծրագրի ցանկացած կետից
- Ռեկուրսիվ կոչվում է այն ֆունկցիան, որը կանչում է ինքն իրեն
- Խնդիրը ռեկուրսիվ լուծելու համար, այն պետք է բաժանել մեկ կամ մի քանի ենթախնդիրների և ավելացնել մեկ կամ մի քանի պայմաններ վերադարձի համար

Ռեկուրսիվ ֆունկցիաներ

- Հաշվել 1-ից n թվերի գումարը

```
#include <iostream>

unsigned sum(unsigned n) {
  if (n == 1) {
    return 1;
  }
  else {
    return n + sum(n-1);
  }
}

int main() {
  std::cout << sum(4);
}
```

https://repl.it/@HaykAslanyan/sumn



Ռեկուրսիվ ֆունկցիաներ

- Հաշվել 1-ից n թվերի գումարը

```
#include <iostream>

unsigned sum(unsigned n) {
  if (n == 1) {
    return 1;
  }
  else {
    return n + sum(n-1);
  }
}

int main() {
  std::cout << sum(4);
}
```

Համարժեք են

```
#include <iostream>

unsigned sum(unsigned n) {
  if (n == 1) {
    return 1;
  }
  return n + sum(n-1);
}

int main() {
  std::cout << sum(4);
}
```

https://repl.it/@HaykAslanyan/sumn

Ռեկուրսիվ ֆունկցիաներ

- n ոչ բացասական ամբողջ թվի ֆակտորիալ.

  $n! = 1 * 2 * 3 * \ldots * n$

  $0! = 1$

```
#include <iostream>
unsigned factorial(unsigned n) {
  if (n <= 1) { // base case
    return 1;
  }
  else {
    return n * factorial(n-1);
  }
}

int main() {
  unsigned fact = factorial(5);
  std::cout << fact;
}
```

https://repl.it/@HaykAslanyan/factorial

Ռեկուրսիվ ֆունկցիաներ

- n ոչ բացասական ամբողջ թվի ֆակտորիալ.

  $n! = 1 * 2 * 3 * \ldots * n$

  $0! = 1$

```
#include <iostream>
unsigned factorial(unsigned n) {
  if (n <= 1) { // base case
    return 1;
  }
  else {
    return n * factorial(n-1);
  }
}

int main() {
  unsigned fact = factorial(5);
  std::cout << fact;
}
```

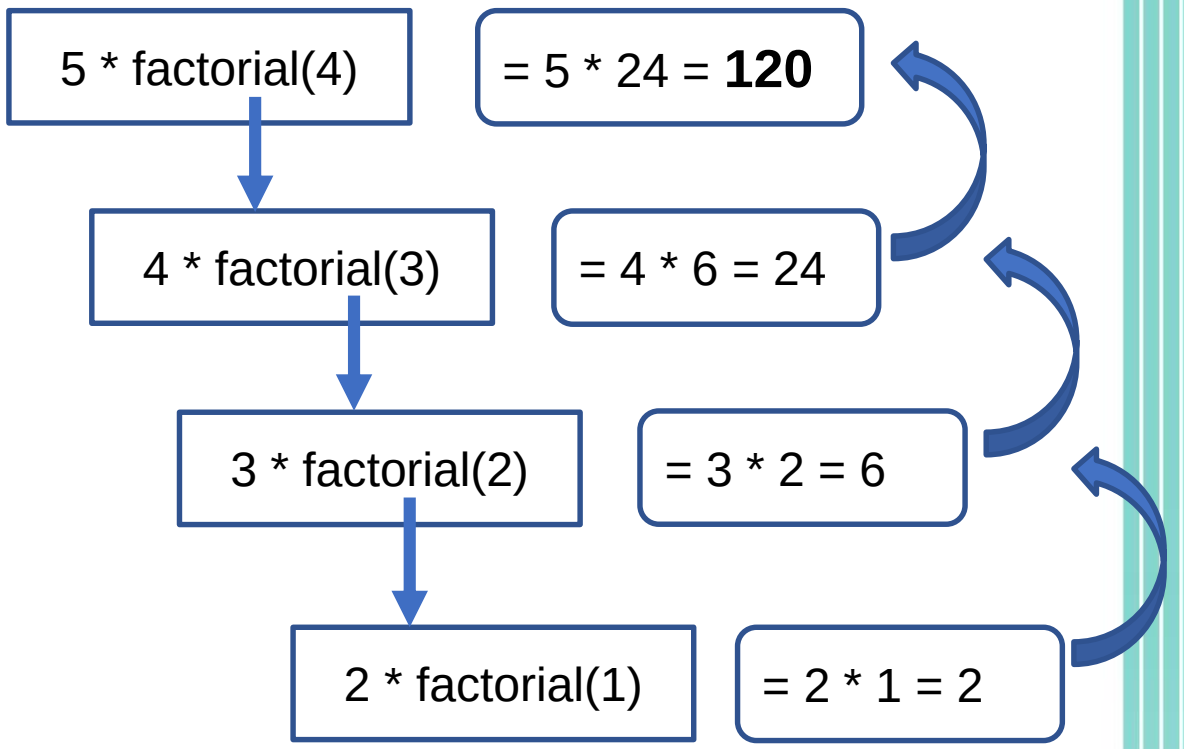
Համարժեք են

```
#include <iostream>
unsigned factorial(unsigned n) {
  if (n <= 1) { // base case
    return 1;
  }
  return n * factorial(n-1);
}

int main() {
  unsigned fact = factorial(5);
  std::cout << fact;
}
```

https://repl.it/@HaykAslanyan/factorial

Ռեկուրսիվ ֆունկցիաներ

```
#include <iostream>
unsigned factorial(unsigned n) {
  if (n <= 1) { // base case
    return 1;
  }
  else {
    return n * factorial(n-1);
  }
}

int main() {
  unsigned fact = factorial(5);
  std::cout << fact;
}
```



https://repl.it/@HaykAslanyan/factorial

Ռեկուրսիվ ֆունկցիաներ

Ֆիբոնաչիի հաջորդականության առաջին երկու թվերն են 0 և 1, իսկ յուրաքանչյուր հաջորդ թիվը հավասար է նախորդ երկու թվերի գումարին

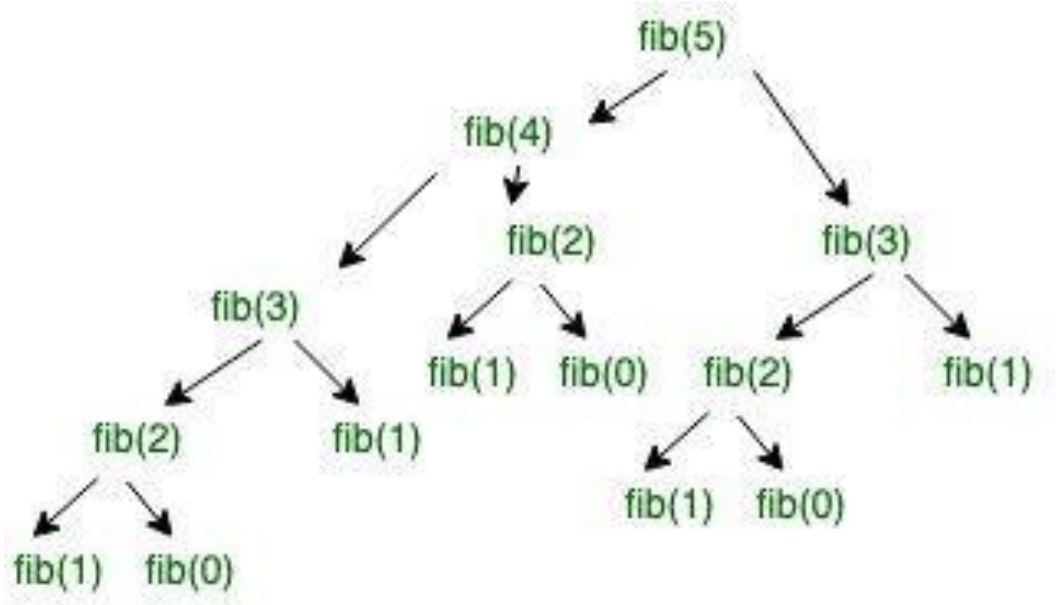
$$F_0 = 0, F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, n \geq 2$$

```
#include <iostream>

int fib(int n) {
  if (n == 0) { // base case
    return 0;
  }
  if (n == 1) { // base case
    return 1;
  }
  return (fib(n - 1) + fib(n - 2));
}
int main() {
  int n = 5;
  std::cout << "Fibonacci series of " << n
<< " numbers is: ";
  for (int i = 0; i <= n; i++) {
    std::cout <<  fib(i) << " ";
  }
}
```

https://repl.it/@HaykAslanyan/fib

Ռեկուրսիվ ֆունկցիաներ




```
#include <iostream>

int fib(int n) {
  if (n == 0) { // base case
    return 0;
  }
  if (n == 1) { // base case
    return 1;
  }
  return (fib(n - 1) + fib(n - 2));
}
int main() {
  int n = 5;
  std::cout << "Fibonacci series of " << n
<< " numbers is: ";
  for (int i = 0; i <= n; i++) {
    std::cout <<  fib(i) << " ";
  }
}
```

https://repl.it/@HaykAslanyan/fib

Խնդիրներ

1. Գրել ծրագիր, որը կտպի առաջին 50 բնական թվերը
2. Գրել ծրագիր , որը կտպի տրված զանգվածի տարրերը
3. Տպել մուտքագրված թվի թվանշանների քանակը
4. Տպել տրված զանգվածի մեծագույն էլեմենտը
5. Մուտքագրել N բնական թիվը և հաշվել 1 + 1/2 + 1/3 + 1/4 … +1/N արտահայտության արժեքը
6. Մուտքագրել N բնական թիվը և հաշվել 1 – 1/2 + 1/3 – 1/4 … 1/N արտահայտության արժեքը

Ռեկուրսիվ ֆունկցիաներ

```
#include <iostream>

int factorial(int n) {
  // wrong base case
  if (n == 100) {
    return 1;
  }
  return n * factorial(n-1);
}

int main() {
  std::cout << factorial(5) << std::endl;
}
```

Անվերջ ռեկուրսիա

https://repl.it/@HaykAslanyan/badrecursion

Ռեկուրսիվ ֆունկցիաներ

Բացահայտ ռեկուրսիա

```
void directRecFun() {
    // ...
    directRecFun();
    // ...
}
```

Ոչ բացահայտ ռեկուրսիա

```
void indirectRecFun1() {
    // ...
    indirectRecFun2();
    // ...
}
void indirectRecFun2() {
    // ...
    indirectRecFun1();
    // ...
}
```

Պոչային ռեկուրսիա

- Պոչային ռեկուրսիայի դեպքում ռեկուրսիվ կանչից հետո ոչ մի հաշվարկ չի կատարվում
- Պոչային ռեկուրսիայի դեպքում կոմպիլյատորը կարող է կատարել օպտիմիզացիա և ռեկուրսիան վերածել ցիկլի
- Պոչային ռեկուրսիայի օպտիմիզացիան ստանդարտով նախատեսված չէ, սակայն դա **որոշ** դեպքերում կատարում են շատ կոմպիլյատորներ – g++, MS Visual Studio, clang++

Պոչային ռեկուրսիա

```
#include <iostream>

void print(int n) {
  if (n < 0) {
   return;
  }
  std::cout << " " << n;

  // The last executed statement is recursive call
  print(n-1);
}

int main() {
  print(50);
}
```

https://repl.it/@HaykAslanyan/tailrecursionprint

Ոչ պոչային ռեկուրսիայի օրինակ

```
#include <iostream>

int fib(int n) {
  if (n == 0) { // base case
    return 0;
  }
  if (n == 1) { // base case
    return 1;
  }
  else {
    return (fib(n - 1) + fib(n - 2));
  }
}
int main() {
  int n = 5;
  std::cout << "Fibonacci series of " << n << " numbers is: ";

  for (int i = 0; i <= n; i++) {
    std::cout <<  fib(i) << " ";
  }
}
```

Այս դեպքում պոչային ռեկուրսիա չէ, վերջին հրամանում 2 կանչ է կատարվում և գումար հաշվվում

https://repl.it/@HaykAslanyan/fib

Ոչ պոչային ռեկուրսիայի օրինակ

```
unsigned factorial(unsigned n) {
  if (n <= 1) { // base case
    return 1;
  }
  else {
    return n * factorial(n-1);
  }
}
```

Այս դեպքում պոչային ռեկուրսիա չէ, վերջին հրամանում 1 կանչ է կատարվում և արտադրյալ հաշվվում

Պոչային ռեկուրսիա

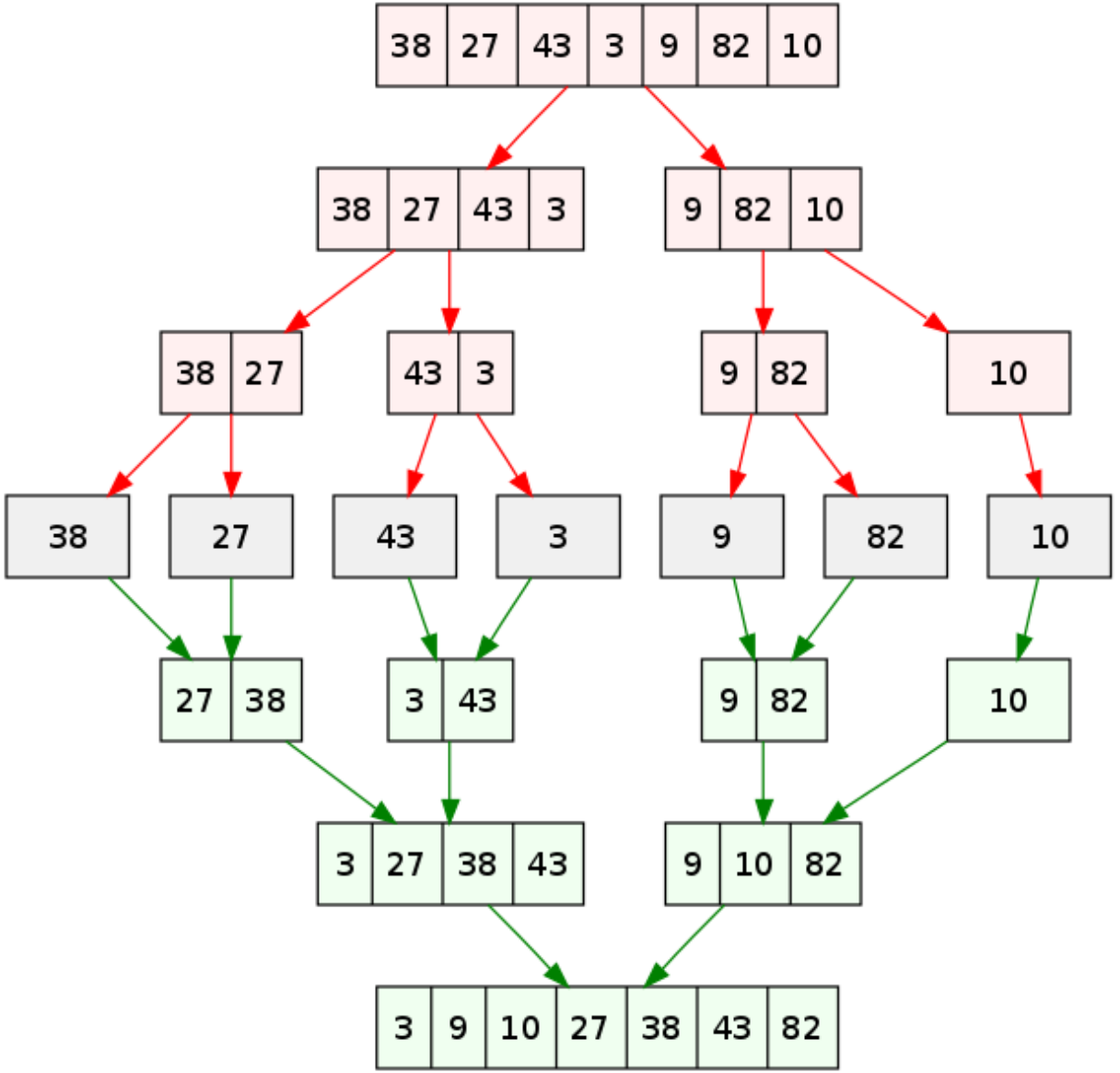
```
#include<iostream>

// A tail recursive function to calculate factorial
unsigned factTR(unsigned n, unsigned a) {
  if (n == 0) {
    return a;
  }
  return factTR(n-1, n*a);
}

unsigned factorial(unsigned n) {
  return factTR(n, 1);
}

int main() {
  std::cout << factorial(5);
  return 0;
}
```

https://repl.it/@HaykAslanyan/tailrecursion



Սորտավորում ձուլման միջոցով

- Օգտագործում է <<բաժանիր որ տիրես>> ռազմավարությունը
- Բաժանում է զանգվածը երկու մասի, սորտավորում 2 մասերը, այնուհետև դրանք ձուլում
- Բարդությունը վատագույն դեպքում - $O(n \log_2 n)$



Սորտավորում ձուլման միջոցով

```
int main() {
  int arr[] = { 12, 11, 13, 5, 6, 7};
  int arr_size = sizeof(arr) / sizeof(arr[0]);

  std::cout << "Given array is \n";
  printArray(arr, arr_size);

  mergeSort(arr, 0, arr_size - 1);

  std::cout << "\nSorted array is \n";
  printArray(arr, arr_size);
}
```

https://repl.it/@HaykAslanyan/mergesort



Սորտավորում ձուլման միջոցով

Սորտավորել զանգվածի *l* ինդեքսից *r* ինդեքս ընկած հատվածը

```
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}

void printArray(int A[], int size) {
  for (int i = 0; i < size; i++) {
    std::cout << A[i] << " ";
  }
  std::cout << "\n";
}
```

https://repl.it/@HaykAslanyan/mergesort



Սորտավորում ձուլման միջոցով

```
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}

void printArray(int A[], int size) {
  for (int i = 0; i < size; i++) {
    std::cout << A[i] << " ";
  }
  std::cout << "\n";
}
```

Սորտավորել զանգվածի *l* ինդեքսից *r* ինդեքս ընկած հատվածը

Սորտավորել նշված հատվածի առաջին կեսը

https://repl.it/@HaykAslanyan/mergesort

Սորտավորում ձուլման միջոցով

```
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}

void printArray(int A[], int size) {
  for (int i = 0; i < size; i++) {
    std::cout << A[i] << " ";
  }
  std::cout << "\n";
}
```

Սորտավորել զանգվածի *l* ինդեքսից *r* ինդեքս ընկած հատվածը

Սորտավորել նշված հատվածի առաջին կեսը

Սորտավորել նշված հատվածի երկրորդ կեսը

https://repl.it/@HaykAslanyan/mergesort

Սորտավորում ձուլման միջոցով

```
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}

void printArray(int A[], int size) {
  for (int i = 0; i < size; i++) {
    std::cout << A[i] << " ";
  }
  std::cout << "\n";
}
```

Սորտավորել զանգվածի *l* ինդեքսից *r* ինդեքս ընկած հատվածը

Սորտավորել նշված հատվածի առաջին կեսը

Սորտավորել նշված հատվածի երկրորդ կեսը

Ձուլել

https://repl.it/@HaykAslanyan/mergesort

Զուլում

```
void merge(int arr[], int l, int m, int r) {
  int i, j, k;
  int n1 = m - l + 1;
  int n2 = r - m;

  int *L = new int[n1], *R = new int[n2];

  for (i = 0; i < n1; i++) {
    L[i] = arr[l + i];
  }
  for (j = 0; j < n2; j++) {
    R[j] = arr[m + 1 + j];
  }

  i = 0;
  j = 0;
  k = l;
//…
```

Զուլել *l*-ից *m* ինդեքսների և *m*-ից *r* ինդեքսների հատվածները

Օգնող զանգվածներ

Հատվածների պատճենում

```
    // ...
    while (i < n1 && j < n2) {
      }
      else {
      }
      k++;
    while (i < n1) {
      arr[k] = L[i];
      i++;
      k++;
    }
    while (j < n2) {
      arr[k] = R[j];
      j++;
      k++;
    }
    delete [] L;
    delete [] R;
}
```

https://repl.it/@HaykAslanyan/mergesort

Զուլում

```
void merge(int arr[], int l, int m, int r) {
  int
  int
  int n2 = r - m;

  int *L = new int[n1], *R = new int[n2];

  for (i = 0; i < n1; i++) {
    L[i] = arr[l + i];
  }
  for (j
    R[j]
  }

  i = 0;
  j = 0;
  k = l;
//…
```

Հատվածների ձուլում

L զանգվածում մնացած էլեմենտների պատճենում

R զանգվածում մնացած էլեմենտների պատճենում

```
// ...
 while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    }
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
 }
 while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
 }
 while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
 }
 delete [] L;
 delete [] R;
}
```

Տնային աշխատանք

Տնային աշխատանք 21-26:

Վարժություններ

- 00 Նախավարժանք
- 01 Թվաբանություն և ճյուղավորում
- 02 Ցիկլեր և ստատիկ զանգվածներ
- 03 Դինամիկ զանգվածներ և ֆունկցիաներ
- 04 Դասեր

Շնորհակալություն. Հարցե՞ր