Таблица

Таблица (множество) е група от елементи от тип T, над които се изпълняват операциите:
- **създаване на празна таблица;**
- **включване – образуване на нова таблица с добавяне на един елемент;**
- **търсене – проверка дали елемент принадлежи на таблицата;**
- **изключване – образуване на нова таблица чрез изключване на даден елемент;**
- **проверка за празна таблица.**

Всеки елемент на таблицата е от тип T и има ключ.

Видове

1. **Последователна неподредена таблица**
2. **Последователна подредена таблица**

Приложение

– бази данни.

Логическо описание

Съединение на елементи от тип T, образуващи тяло на таблицата.

данни: тип T

тип ТАБЛИЦА$_T$ = (ПРАЗНО | НЕПРАЗНА_ТАБЛИЦА$_T$)

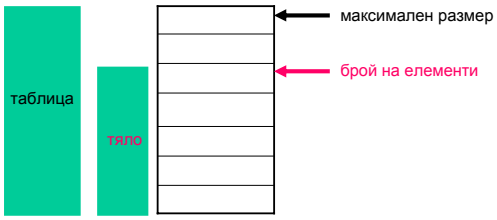
тип НЕПРАЗНА_ТАБЛИЦА$_T$ = (тяло: ТАБЛИЦА$_T$)

Операции

1. **Създаване на таблица**
2. **Празна таблица**
3. **Включване на елемент в таблица**
4. **Търсене на елемент в таблица**
5. **Изключване на елемент от таблица**

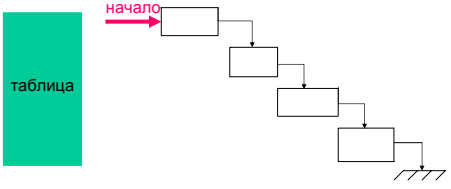
Физическо представяне

1. **Непрекъснато представяне**



тяло – **фиксирана област от паметта (масив)**
брой – **тип** ЦЯЛО

2. **Верижно представяне**



Дефиниране

```
#define SIZE <константа>        // максимален размер
typedef <тип> DATA;             // тип на данни
struct set                      // таблица
{
  int n;                        // брой елементи в таблицата
  DATA key[SIZE];               // тяло
};
typedef struct set SET;

#define FAILURE 0               // успешна операция
#define SUCCESS 1               // неуспешна операция
```

Последователна неподредена таблица – характеристики

Добавяне на елемент в неподредена таблица – в края на таблицата.

Изключване на елемент в неподредена таблица – последният елемент от таблицата се записва на мястото на изключения елемент.

1. Създаване на празна таблица

Алгоритъм

брой ← **ПРАЗНО**

```
SET create()
{
  SET s;
  s.n = 0;                      // празна таблица
  return s;
}
```

2. Празна таблица

Алгоритъм

ако брой **= ПРАЗНО**
  таблицата е празна
в противен случай
  таблицата не е празна

```
                    таблица
int empty (SET s)
{
  return !s.n;
}
```

3. Включване на елемент в таблица

Алгоритъм

ако има място в таблицата
  брой ← брой **+ 1**
  тяло**[**брой**]** ← **елемент**
  успешна операция
в противен случай
  неуспешна операция

```
        елемент                 таблица
int insert (DATA x, SET *s)
{
  if (s->n < SIZE-1)
  {
    (s->n)++;
    s->key[s->n] = x;
    return SUCCESS;
  }
  else
    return FAILURE;
}
```

4. Търсене на елемент в таблица

Алгоритъм

ако таблицата е празна
  неуспешна операция
в противен случай
  за i от 1 до брой **стъпка 1**
  повтаряй
    ако ключ(елемент$_i$) = ключ(елемент)
      индекс ← I
      успешна операция
  неуспешна операция

```
int search (DATA key, SET s, int *index)
{  int i;
   if (empty (s))
   {
      printf("Празна таблица!\n");
      return FAILURE;
   }
   else
   {
       for (i=1; i<=s.n; i++)
          if (s.key[i].ключ == key.ключ)
          {
             *index = i;
             return SUCCESS;
          }
         return FAILURE;
   }
}
```

индекс на намерения елемент

таблица

търсен елемент с даден ключ

5. Изключване на елемент от таблица

Алгоритъм

ако елементът за изключване е намерен
  последният елемент се записва на мястото на намерения елемент
  брой ← брой **- 1**
  успешна операция
неуспешна операция

```
int del (DATA x, SET *s)
{
   int i;
   if (search (x, *s, &i) == SUCCESS)
   {
      s->key[i] = s->key[s->n];
      (s->n)--;
      return SUCCESS;
   }
   return FAILURE;
}
```

елемент

таблица

индекс на намерения елемент

Последователна подредена таблица – характеристики

Добавяне на елемент в подредена таблица – в място, за което ключовете на предшестващите елементи са по-малки от ключа на добавения елемент.

Изключване на елемент в подредена таблица – последователно преместване на елементите след изключения с едно място напред.

1. Създаване на празна таблица

Алгоритъм

брой ← **ПРАЗНО**

```
SET create()
{
   SET s;
   s.n = 0;                    // празна таблица
   return s;
}
```

2. Празна таблица

Алгоритъм

ако брой **= ПРАЗНО**
  таблицата е празна
в противен случай
  таблицата не е празна

таблица

```
int empty (SET s)
{
  return !s.n;
}
```

3. Включване на елемент в таблица

Алгоритъм

ако има място в таблицата
  за i=брой **до i>=1 и ключ(елемент)<ключ(елемент$_i$)**
  стъпка -1 повтаряй
    елемент$_{i+1}$ ← елемент$_i$
  тяло**[i+1] ← елемент**
  брой **←** брой **+ 1**
  успешна операция
в противен случай
  неуспешна операция

елемент таблица

```
int insert (DATA x, SET *s)
{
  int i;
  if (s->n < SIZE-1)
  {
    for (i = s->n; i >= 1 && x < s->key[i]; i--)
      s->key[i+1] = s->key[i];
    s->key[i+1] = x;
    (s->n)++;
    return SUCCESS;
  }
    return FAILURE;
}
```

4. Търсене на елемент в таблица – двоично търсене

Алгоритъм

ако таблицата е празна
  неуспешна операция
в противен случай
  ляво ← 1
  дясно ← брой
  докато ляво <= дясно
  повтаряй
    среда ← (ляво+дясно) / 2
    ако ключ(елемент) < ключ(елемент$_{среда}$)
      дясно ← среда - 1
    в противен случай
      ако ключ(елемент) > ключ(елемент$_{среда}$)
        ляво ← среда + 1
      в противен случай
        индекс ← среда
        успешна операция
  неуспешна операция

```
int search (DATA key, SET s, int *index)
{  int left, right, middle;
   if (empty (s))
   {  printf("Празно множество!\n");
      return FAILURE;
   }
   else
   {  left = 1; right = s.n;
      while (left <= right)
      {  middle = (left + right)/2;
         if (key.ключ < s.key[middle].ключ)
            right = middle - 1;
         else if (key.ключ > s.key[middle].ключ)
            left = middle + 1;
         else
         {  *index = middle;
            return SUCCESS;
         }
      }
   }
   return FAILURE;
}
```

индекс на намерения елемент

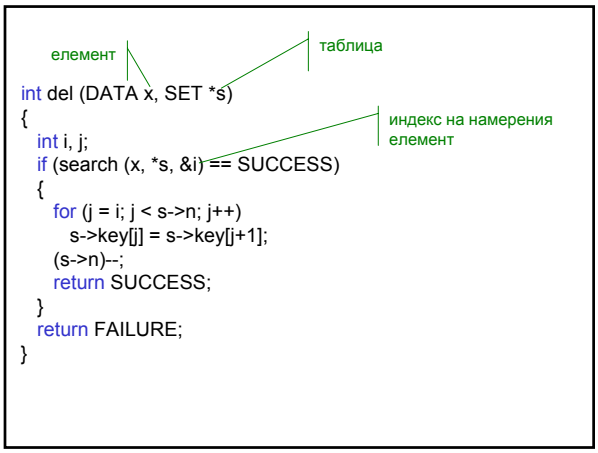
таблица

търсен елемент с даден ключ

5. Изключване на елемент от таблица

Алгоритъм

ако елементът за изключване е намерен
  за j=намереното място до брой **стъпка 1**
  повтаряй
    елемент$_j$ ← елемент$_{j+1}$
  брой **←** брой **- 1**
  успешна операция
неуспешна операция




```
int del (DATA x, SET *s)
{
  int i, j;
  if (search (x, *s, &i) == SUCCESS)
  {
    for (j = i; j < s->n; j++)
      s->key[j] = s->key[j+1];
    (s->n)--;
    return SUCCESS;
  }
  return FAILURE;
}
```

Задача: **Структура от данни** ТАБЛИЦА

Неподредена таблица, съдържаща информация за сметките (структура от данни СМЕТКА**) на вложителите в банка.**

1. **Напишете функция** print()**, която отпечатва множеството от вложители в банка.**

Алгоритъм

```
функция печат: (аргумент s: МНОЖЕСТВО)
  променлива i
  ако s е ПРАЗНО
    отпечатай Празно множество
  в противен случай
    за i=1 до брой стъпка 1
    повтаряй
      печат(сметка)
```

2. **Използвайте функциите:**
   - create() **за създаване на неподредена таблица с информация за сметките на вложители в банка;**
   - insert() **за включване на сметки на вложители в таблицата;**
   - search() **за търсене на дадена сметка в таблицата;**
   - del() **за изключване на дадена сметка от таблицата;**
   - **разпечатайте създадената таблица и получените резултати от извикването на горните функции, като използвате функцията** print()**, където е необходимо.**
3. **Реализирайте същата обработка за подредена таблица от сметки на вложители в банка.**