# Anagrafica

Vincenzo Petrolo

November 18, 2019

# Chapter 1

# Data Structures

## 1.1 Minor data structures

```
typedef enum {r_acquisizione,r_codice,r_cancella,r_stampa,r_fine} decisioni;
typedef enum {FALSE,TRUE} boolean;
typedef enum {FILE_TESTO,STDIN,STDOUT} sorgente;
```

### 1.1.1 Prototypes

```
boolean richiedi_cancellazione();
```

## 1.2 Date data structure

```
typedef struct {
int gg;
int mm;
int aaaa;
}data_t;
typedef data_t KEY_1;
```

### 1.2.1 Prototypes

```
boolean data_gt(data_t data1,data_t data2);
boolean KEY_1eq(KEY_1 data_1,KEY_1 data_2);
```

## 1.3 Node data structure

### 1.3.1 Node payload

```
typedef char KEY;

typedef struct {
KEY codice[__NCODICE__];
char nome[__MAX_CHARS__];
char cognome[__MAX_CHARS__];
data_t data;
char via[__MAX_CHARS__];
char citta[__MAX_CHARS__];
int cap;
}Item;
```

### 1.3.2 Simple node structure

```
typedef  struct nodo* link;

struct nodo {
Item val;
link next;
};
```

## 1.4 Prototypes

```
link newNode (Item val, link next);
link SortInsList(link h,Item val);
link getKEY_1(link head,KEY_1 data_2);
link getKEY(link head,KEY *codice);
link node_deleteKey(link h,KEY *k);
void node_display(link node,FILE *fp);
void node_extract(link* head,link node);
void acquisisci_Item(link *head);
decisioni acquisisci_menu(link *head);
boolean KEY_eq(KEY *k1,KEY *k2);
```

# Chapter 2

# Functions

## 2.1 Functions description

### 2.1.1 Minor data structures

**richiedi_cancellazione**

```
boolean richiedi_cancellazione();
```

### 2.1.2 Date structure

This function requests from stdin if the user wants to delete the result of a research.

**data_gt**

```
boolean data_gt(data_t data1,data_t data2);
```

This function return TRUE if the first argument is a newer date than the other,otherwise it return FALSE.

**KEY_1eq**

```
boolean KEY_1eq(KEY_1 data_1,KEY_1 data_2);
```

This function return TRUE if the first argument is equal to the second,otherwise it return FALSE.

### 2.1.3 Node structure

#### newNode

```
link newNode (Item val, link next);
```

This function creates a Node and returns the pointer to it.

#### SortInsList

```
link SortInsList(link h,Item val);
```

This function inserts a node generated with the newNode function, and fill it with the Item val parameter. Then referring to the given order to the list, it puts the node in the right position.It returns the pointer to the updated list.

#### getKEY_1

```
link getKEY_1(link head,KEY_1 data_2);
```

This function extracts from the given list the first occurrence of the key passed as argument to the functions. It returns the pointer to the node.

#### getKEY

```
link getKEY(link head,KEY *codice);
```

This function is the same as the one mentioned above, but it extracts a KEY type different.

#### node_deleteKey

```
link node_deleteKey(link h,KEY *k);
```

This function is used to delete the first occurrence of a node which has the same KEY as the one passed by value to the function.

#### node_display

```
void node_display(link node,FILE *fp);
```

This function is used to display the output of the given node. The output can be either stdout or a FILE *.

**node_extract_print**

```
void node_extract_print(link* head,link node);
```

This function is used to extract the given node and print its output, it can be used to avoid using the two functions mentioned above.

**acquisisci_Item**

```
void acquisisci_Item(link *head);
```

This function is used to read an Item type. It stores the read value in a new node which is linked to the given list.

**acquisisci_menu**

```
decisioni acquisisci_menu(link *head);
```

This is the main calling function,it prints in stdout the menu, and the user can make choices from it.From here starts every function call.

**KEY_eq**

```
boolean KEY_eq(KEY *k1,KEY *k2);
```

This function has the purpose to compare two KEY Item, it returns TRUE if the two value are the same.