**Discipline** - SoftwareEngineeringMethods.

Laboratory №8-9

**The Method of software engineering oriented on data structures**

**Jackson strategy**

Tool design: MicrosoftVisio

Contents

1. Statement of a problem
2. Requirements Analysis
3. Design of Diagrams
4. Testing
5. Conclusion

**Introduction**

Jackson structured programming (JSP) is a method for structured programming based on correspondences between data stream structure and program structure. JSP structures programs and data in terms of sequences, iterations and selections, and as a consequence, it is applied when designing a program's detailed control structure, below the level where object-oriented methods become important.

JSP uses semi-formal steps to capture the existing structure of a program's inputs and outputs in the structure of the program itself.

The intent is to create programs which are easy to modify over their lifetime. Jackson's major insight was that requirement changes are usually minor tweaks to the existing structures. For a program constructed using JSP, the inputs, the outputs, and the internal structures of the program all match, so small changes to the inputs and outputs should translate into small changes to the program.

JSP structures programs in terms of four component types:

* fundamental operations
* sequences
* iterations
* selections

The method begins by describing a program's inputs in terms of the four fundamental component types. It then goes on to describe the program's outputs in the same way. Each input and output is modelled as a separate Data Structure Diagram (DSD). To make JSP work for compute-intensive applications, such as digital signal processing (DSP) it is also necessary to draw algorithm structure diagrams, which focus on internal data structures rather than input and output ones.

The input and output structures are then unified or merged into a final program structure, known as a Program Structure Diagram (PSD). This step may involve the addition of a small amount of high level control structure to marry up the inputs and outputs. Some programs process all the input before doing any output, whilst others read in one record, write one record and iterate. Such approaches have to be captured in the PSD.

The PSD, which is language neutral, is then implemented in a programming language. JSP is geared towards programming at the level of control structures, so the implemented designs use just primitive operations, sequences, iterations and selections. JSP is not used to structure programs at the level of classes and objects, although it can helpfully structure control flow within a class's methods.

JSP uses a diagramming notation to describe the structure of inputs, outputs and programs, with diagram elements for each of the fundamental component types.

**Task declaration**

Purpose of this laboratory work – is create program for working with these:

* Stack
* Queue
* Linked list
* Double linked list
* Cycled double linked list

They all must be functions and have sub functions. For example these: function “stack” that have sub function “write” or “delete”.

Must be realize these functions with sub functions:

* Stack
  + Create;
  + Insert;
  + Delete;
  + Search element by key;
  + Search position by entered value;
  + Count quantity of elements in stack;
  + Clean stack elements;
  + Search of max element.
* Queue
* Write
* Output
* Delete
* Count quantity of elements(size function)
* Clean queue
* Search by value
* Double linked list
  + Creating
  + Adding in end position
  + Adding in start position
  + Search element by key and value
  + Adding element by entering position
  + Deleting element
  + Output
  + Clean

**Classification of linked line lists**



**Problem statement**

For creating program, we need to:

Consider ways of organizing the data structure on the example of linear lists associated with the use of Jackson's chart. Decompose program to two modules:

* Single linked lists
* Double linked lists

Decompose modules to unary functions.

* Stack
* Queue
* Linear list
* Cycled(ringed)list
* Double linked list

Consider unary functions and decompose them to the sub unary functions (at the prev. page).

Then we need to:

* Create flowcharts in Microsoft Visio 2013;
* Write program code in Microsoft Visual C++ 2013;
* Test unary function;
* Test sub unary functions;
* Test program for unique moments.

**Consider operation based on a linear list stack.**

The stack works on the principle: «last in, first out» or last came in, first out. Remove and add an item to the stack runs only top of the list. An example of this is the stack book: to take a third book-you need to take the book to the third.

Decompose function to modules:

-create/output;

-insert/delete;

-search;

-count;

-clean;

Decompose modules to unary functions.

Also, need to:

-Create a structure struct stack {int data; //inform field

stack \*next; // field address};

-create global variables with values: stack \*head=NULL; stack \*now=NULL; int n=0;

-create functions input(for input data in stack); out(for output data from stack);deleete(for delete data from stack); search\_bv(for search element by value in stack); search\_ba(for search element by address in stack); check\_st(for checking a number of elements in stack); clean\_st(for clean all data in a stack); mandm(for search max and min elements in stack).

For input function : int count;//number of inputted elements.

For deleete function int dt\_el;//saving the value of deleting element.

For search\_ba function int address;//for saving address of searching element.

For mandm function int max;int min;//for saving max and min values.

**Decomposition of the main function.**



**Main function decompose**

**Linear linked list**

In linear lists already described connection between the elements in this case is related to the following elements. Delete an item passes through the removal of one element connection, and this connection is routed to the next item, which is located behind it.

Adding an item passes through the removal of connection of one element to another and this connection is assigned to the new element, and that in turn builds a relationship with the next element.

In linear lists already, described connection between the elements in this case is relate to the following elements. Delete an item passes through the removal of one element connection, and this connection is route to the next item, which is located behind it.

Adding an item passes through the removal of connection of one element to another and this connection is assign to the new element and that in turn builds a relationship with the next element







The stack works by a principle «last in, first out». The queue works by a principle : «First in, first out».









**Double linked list**

Double linked list, this same sequence of elements only, unlike them, they have no connection with the previous elements and follow them. In other words, the son knows his ancestor and predecessor, who in turn knows its child and parent.

Deleting an element in the doubly linked list. The doubly linked lists are deleted pointers to the desired item, and so as not to disrupt the relationship of these communications, you must link to the previous and next element, standing before and after the deleted

Adding an element to a doubly linked list. Removed those links, to whom the new element, and these connections will be insert to connect the new element, and that in turn binds to the next list item





**Ringed double linked list**

Ring list - this is the list, which has a connection with the first and last item in the list.

Deleting a -Removes pointer to the desired item, and not to disturb the relationship, these relationships need to be associated with the next and previous element, and before standing after being remove. If you delete the item passes from the beginning or end of the list, you need to pass the next or previous element of those signs that have been remove from the element.

Adding element- removed those links, to whom the new element, and these connections will be insert to connect the new element, and that in turn is associated with the following items in the list. When adding an item passes from the beginning or end of the list, you need to pass the new element are pointers which were in the former initial or final element.









**Example - Procedure for removing an initial element of a doubly linked circular list**

PV=Head

Head=Head->next

Head->prev=Last

Last->next=Head

Last->prev->next=Last

DeletePV

**CONCLUSION**

During a lab technique of designing Jackson was investigated. As mentioned in the introduction, the basis of this method is based on the basic idea of ​​structured programming Jackson. Three main stages were considered:

• Modeling Stage;

• Newtwork-stage;

• Implementation phase.

Considered structured programming constructs that Jackson methodology used to construct the input and output data, which are then used to build the program. These are the following constructions:

• The design of the sequence;

• choice of design;

• repetition of design.

It was found that the area of ​​concern for any elements of the system are the threads, processes and data structures. In the structural analysis are active only with the data flows and processes. Thus, methods oriented data structure, provides:

• Identification of key information objects and operations;

• definition of a hierarchical data structure;

• layout structures of standard data structures are sequences, selection, repetition;

• sequence of steps to transform the hierarchical structure