Name: Farhad Ali
Reg No: SP23-BSE-054
Subject: DS theory
Assignment No: 01

LOGIC OF CODE:

Logic we used in code.

· 1 Task Node Structure:

- Each task is represented as a node containing the following:
- taskID: A unique identifier for the task.
- description: Details of the task.
- priority: A numeric value representing the importance of the task (higher numbers indicate higher priority).
- next: A pointer to the next node (task) in the list.

· 2 Adding Tasks:

- When a new task is added, the list is traversed to find the correct position based on its priority.
- The task is inserted either at the beginning (if it has the highest priority) or in the appropriate position where tasks with higher priority come before it.

·3 Viewing Tasks:

- The function traverses the entire list starting from the head (first node) and displays the task details of each node.
- If the list is empty, a message indicating "No tasks available" is displayed.

· 4 Removing the Highest Priority Task:

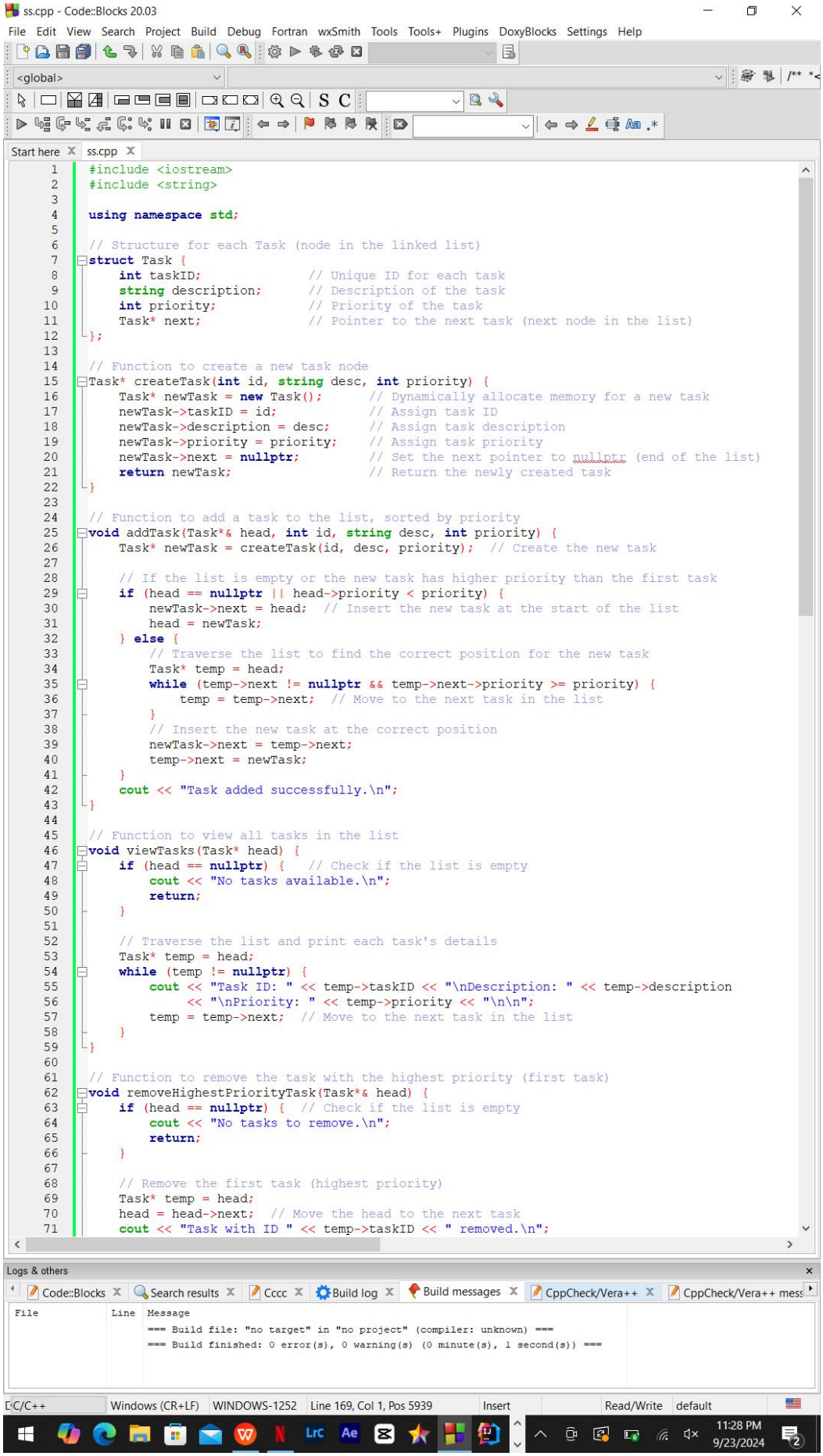
- Since the list is sorted by priority, the task with the highest priority is always the first node (head of the list).
- The head pointer is moved to the next task, and the previous first task is deleted from memory.

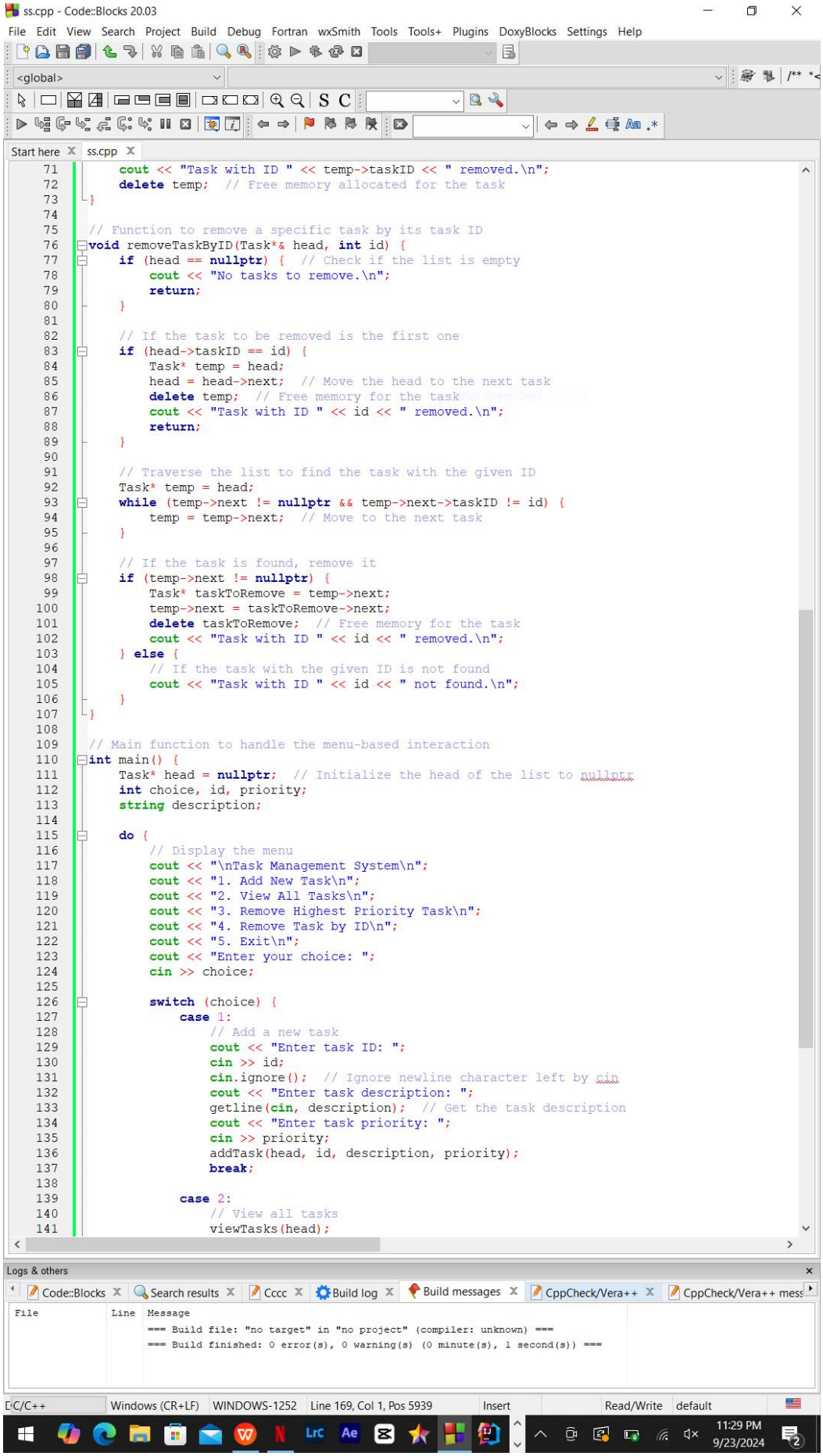
· 5 Removing a Task by ID:

- The list is traversed to find the task with the given taskID.
- Once found, the task is removed by adjusting the pointers of the previous node to skip over the node to be deleted.
- If the task is not found, an error message is shown.

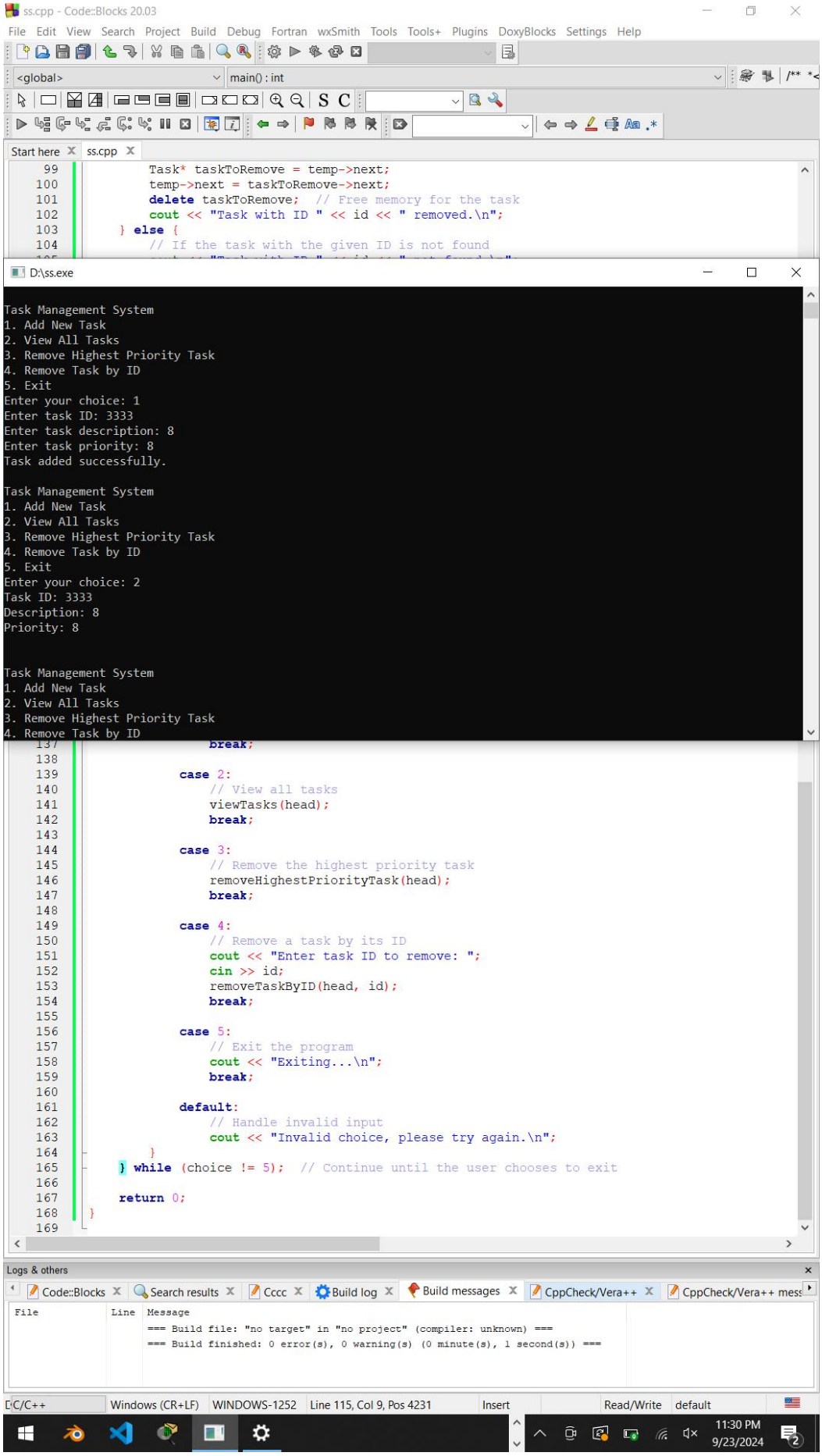
·6 Main Menu Interaction:

- A loop presents a menu to the user, allowing them to add tasks, view all tasks, remove tasks by ID or highest priority, or exit the system.
- The program continues prompting the user until they choose the exit option.


ss.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
ss.cpp
#include <iostream>
#include <string>
using namespace std;
// Structure for each Task (node in the linked list)
struct Task {
int taskID; // Unique ID for each task
string description; // Description of the task
int priority; // Priority of the task
Task* next; // Pointer to the next task (next node in the list)
};
// Function to create a new task node
Task* createTask(int id, string desc, int priority) {
Task* newTask = new Task(); // Dynamically allocate memory for a new task
newTask->taskID = id; // Assign task ID
newTask->description = desc; // Assign task description
newTask->priority = priority; // Assign task priority
newTask->next = nullptr; // Set the next pointer to nullptr (end of the list)
return newTask; // Return the newly created task
}
// Function to add a task to the list, sorted by priority
void addTask(Task*& head, int id, string desc, int priority) {
Task* newTask = createTask(id, desc, priority); // Create the new task
// If the list is empty or the new task has higher priority than the first task
if (head == nullptr || head->priority < priority) {
newTask->next = head; // Insert the new task at the start of the list
head = newTask;
} else {
// Traverse the list to find the correct position for the new task
Task* temp = head;
while (temp->next != nullptr && temp->next->priority >= priority) {
temp = temp->next; // Move to the next task in the list
}
// Insert the new task at the correct position
newTask->next = temp->next;
temp->next = newTask;
}
cout << "Task added successfully.\n";
}
// Function to view all tasks in the list
void viewTasks(Task* head) {
if (head == nullptr) { // Check if the list is empty
cout << "No tasks available.\n";
return;
}
// Traverse the list and print each task's details
Task* temp = head;
while (temp != nullptr) {
cout << "Task ID: " << temp->taskID << "\nDescription: " << temp->description
<< "\nPriority: " << temp->priority << "\n\n";
temp = temp->next; // Move to the next task in the list
}
}
// Function to remove the task with the highest priority (first task)
void removeHighestPriorityTask(Task*& head) {
if (head == nullptr) { // Check if the list is empty
cout << "No tasks to remove.\n";
return;
}
// Remove the first task (highest priority)
Task* temp = head;
head = head->next; // Move the head to the next task
cout << "Task with ID " << temp->taskID << " removed.\n";
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ mess
File
Line
Message
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 169, Col 1, Pos 5939
Insert
Read/Write
default
11:28 PM
9/23/2024


ss.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
ss.cpp
cout << "Task with ID " << temp->taskID << " removed.\n";
delete temp; // Free memory allocated for the task
}
// Function to remove a specific task by its task ID
void removeTaskByID(Task*& head, int id) {
if (head == nullptr) { // Check if the list is empty
cout << "No tasks to remove.\n";
return;
}
// If the task to be removed is the first one
if (head->taskID == id) {
Task* temp = head;
head = head->next; // Move the head to the next task
delete temp; // Free memory for the task
cout << "Task with ID " << id << " removed.\n";
return;
}
// Traverse the list to find the task with the given ID
Task* temp = head;
while (temp->next != nullptr && temp->next->taskID != id) {
temp = temp->next; // Move to the next task
}
// If the task is found, remove it
if (temp->next != nullptr) {
Task* taskToRemove = temp->next;
temp->next = taskToRemove->next;
delete taskToRemove; // Free memory for the task
cout << "Task with ID " << id << " removed.\n";
} else {
// If the task with the given ID is not found
cout << "Task with ID " << id << " not found.\n";
}
}
// Main function to handle the menu-based interaction
int main() {
Task* head = nullptr; // Initialize the head of the list to nullptr
int choice, id, priority;
string description;
do {
// Display the menu
cout << "\nTask Management System\n";
cout << "1. Add New Task\n";
cout << "2. View All Tasks\n";
cout << "3. Remove Highest Priority Task\n";
cout << "4. Remove Task by ID\n";
cout << "5. Exit\n";
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
case 1:
// Add a new task
cout << "Enter task ID: ";
cin >> id;
cin.ignore(); // Ignore newline character left by cin
cout << "Enter task description: ";
getline(cin, description); // Get the task description
cout << "Enter task priority: ";
cin >> priority;
addTask(head, id, description, priority);
break;
case 2:
// View all tasks
viewTasks(head);
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
File
Line
Message
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 169, Col 1, Pos 5939
Insert
Read/Write
default
11:29 PM
9/23/2024


ss.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
Start here
ss.cpp
99 Task* taskToRemove = temp->next;
100 temp->next = taskToRemove->next;
101 delete taskToRemove; // Free memory for the task
102 cout << "Task with ID " << id << " removed.\n";
103 } else {
104 // If the task with the given ID is not found
105 cout << "Task with ID " << id << " not found.\n";
106 }
107 }
108
109 // Main function to handle the menu-based interaction
110 int main() {
111 Task* head = nullptr; // Initialize the head of the list to nullptr
112 int choice, id, priority;
113 string description;
114
115 do {
116 // Display the menu
117 cout << "\nTask Management System\n";
118 cout << "1. Add New Task\n";
119 cout << "2. View All Tasks\n";
120 cout << "3. Remove Highest Priority Task\n";
121 cout << "4. Remove Task by ID\n";
122 cout << "5. Exit\n";
123 cout << "Enter your choice: ";
124 cin >> choice;
125
126 switch (choice) {
127 case 1:
128 // Add a new task
129 cout << "Enter task ID: ";
130 cin >> id;
131 cin.ignore(); // Ignore newline character left by cin
132 cout << "Enter task description: ";
133 getline(cin, description); // Get the task description
134 cout << "Enter task priority: ";
135 cin >> priority;
136 addTask(head, id, description, priority);
137 break;
138
139 case 2:
140 // View all tasks
141 viewTasks(head);
142 break;
143
144 case 3:
145 // Remove the highest priority task
146 removeHighestPriorityTask(head);
147 break;
148
149 case 4:
150 // Remove a task by its ID
151 cout << "Enter task ID to remove: ";
152 cin >> id;
153 removeTaskByID(head, id);
154 break;
155
156 case 5:
157 // Exit the program
158 cout << "Exiting...\n";
159 break;
160
161 default:
162 // Handle invalid input
163 cout << "Invalid choice, please try again.\n";
164 }
165 } while (choice != 5); // Continue until the user chooses to exit
166
167 return 0;
168 }
169
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ mess
File
Line
Message
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 115, Col 9, Pos 4231
Insert
Read/Write
default
11:29 PM
9/23/2024


ss.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
ss.cpp
Task* taskToRemove = temp->next;
temp->next = taskToRemove->next;
delete taskToRemove; // Free memory for the task
cout << "Task with ID " << id << " removed.\n";
} else {
// If the task with the given ID is not found
D:\ss.exe
Task Management System
1. Add New Task
2. View All Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
5. Exit
Enter your choice: 1
Enter task ID: 3333
Enter task description: 8
Enter task priority: 8
Task added successfully.
Task Management System
1. Add New Task
2. View All Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
5. Exit
Enter your choice: 2
Task ID: 3333
Description: 8
Priority: 8
Task Management System
1. Add New Task
2. View All Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
break;
case 2:
// View all tasks
viewTasks(head);
break;
case 3:
// Remove the highest priority task
removeHighestPriorityTask(head);
break;
case 4:
// Remove a task by its ID
cout << "Enter task ID to remove: ";
cin >> id;
removeTaskByID(head, id);
break;
case 5:
// Exit the program
cout << "Exiting...\n";
break;
default:
// Handle invalid input
cout << "Invalid choice, please try again.\n";
} while (choice != 5); // Continue until the user chooses to exit
return 0;
}
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
File
Line
Message
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 115, Col 9, Pos 4231
Insert
Read/Write
default
11:30 PM
9/23/2024